Simulación de Sistemas Continuos y a Tramos

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

26 de junio 2007

Buscando Métodos de Simulación para Sistemas Rígidos

Un polinomio de interpolación de orden n puede escribirse:

$$p(s) = a_0 + a_1\ s + a_2\ s^2 + a_3\ s^3 + \cdots + a_n\ s^n$$

donde s es el *tiempo normalizado* introducido antes.

Buscando Métodos de Simulación para Sistemas Rígidos

Un polinomio de interpolación de orden n puede escribirse:

$$p(s) = a_0 + a_1\ s + a_2\ s^2 + a_3\ s^3 + \cdots + a_n\ s^n$$

donde s es el *tiempo normalizado* introducido antes.

Su derivada con respecto al tiempo t puede formularse:

$$h \cdot \dot{p}(s) = a_1 + 2a_2\ s + 3a_3\ s^2 + \cdots + n\ a_n\ s^{n-1}$$

Buscando Métodos de Simulación para Sistemas Rígidos

Un polinomio de interpolación de orden n puede escribirse:

$$p(s) = a_0 + a_1\ s + a_2\ s^2 + a_3\ s^3 + \cdots + a_n\ s^n$$

donde s es el *tiempo normalizado* introducido antes.

Su derivada con respecto al tiempo t puede formularse:

$$h \cdot \dot{p}(s) = a_1 + 2a_2\ s + 3a_3\ s^2 + \cdots + n\ a_n\ s^{n-1}$$

En el caso del algoritmo BDF3 ($n = 3$) sabemos:

$$\begin{aligned} h \cdot \dot{p}(s = +1) &= h \cdot f_{k+1} \\ p(s = 0) &= x_k \\ p(s = -1) &= x_{k-1} \\ p(s = -2) &= x_{k-2} \end{aligned}$$

Buscando Métodos de Simulación para Sistemas Rígidos

Un polinomio de interpolación de orden n puede escribirse:

$$p(s) = a_0 + a_1\ s + a_2\ s^2 + a_3\ s^3 + \cdots + a_n\ s^n$$

donde s es el *tiempo normalizado* introducido antes.

Su derivada con respecto al tiempo t puede formularse:

$$h \cdot \dot{p}(s) = a_1 + 2a_2\ s + 3a_3\ s^2 + \cdots + n\ a_n\ s^{n-1}$$

En el caso del algoritmo BDF3 ($n = 3$) sabemos:

$$\begin{aligned} h \cdot \dot{p}(s = +1) &= h \cdot f_{k+1} \\ p(s = 0) &= x_k \\ p(s = -1) &= x_{k-1} \\ p(s = -2) &= x_{k-2} \end{aligned}$$

Entonces:

$$\begin{aligned} h \cdot f_{k+1} &= a_1 + 2a_2 + 3a_3 \\ x_k &= a_0 \\ x_{k-1} &= a_0 - a_1 + a_2 - a_3 \\ x_{k-2} &= a_0 - 2a_1 + 4a_2 - 8a_3 \end{aligned}$$

Buscando Métodos de Simulación para Sistemas Rígidos II

Es decir:

$$\begin{pmatrix} h \cdot f_{k+1} \\ x_k \\ x_{k-1} \\ x_{k-2} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 \\ 1 & -2 & 4 & -8 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Buscando Métodos de Simulación para Sistemas Rígidos II

Es decir:

$$\begin{pmatrix} h \cdot f_{k+1} \\ x_k \\ x_{k-1} \\ x_{k-2} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 \\ 1 & -2 & 4 & -8 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Podemos determinar los valores de los parámetros y calcular:

$$x_{k+1} = p(s = +1) = a_0 + a_1 + a_2 + a_3$$

Buscando Métodos de Simulación para Sistemas Rígidos II

Es decir:

$$\begin{pmatrix} h \cdot f_{k+1} \\ x_k \\ x_{k-1} \\ x_{k-2} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 \\ 1 & -2 & 4 & -8 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Podemos determinar los valores de los parámetros y calcular:

$$x_{k+1} = p(s = +1) = a_0 + a_1 + a_2 + a_3$$

Haciéndolo de forma simbólica obtenemos:

$$x_{k+1} = \frac{6}{11} h \cdot f_{k+1} + \frac{18}{11} x_k - \frac{9}{11} x_{k-1} + \frac{2}{11} x_{k-2}$$

los coeficientes del algoritmo BDF3 ya derivados antes de otra manera.

Buscando Métodos de Simulación para Sistemas Rígidos II

Es decir:

$$\begin{pmatrix} h \cdot f_{k+1} \\ x_k \\ x_{k-1} \\ x_{k-2} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 \\ 1 & -2 & 4 & -8 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Podemos determinar los valores de los parámetros y calcular:

$$x_{k+1} = p(s = +1) = a_0 + a_1 + a_2 + a_3$$

Haciéndolo de forma simbólica obtenemos:

$$x_{k+1} = \frac{6}{11} h \cdot f_{k+1} + \frac{18}{11} x_k - \frac{9}{11} x_{k-1} + \frac{2}{11} x_{k-2}$$

los coeficientes del algoritmo BDF3 ya derivados antes de otra manera.

Podemos servirnos de este nuevo camino para buscar *otros algoritmos lineales de simulación numérica en múltiples pasos*.

Buscando Métodos de Simulación para Sistemas Rígidos III

Una razón por la cual no funcionó muy bien la extrapolación era a causa de las *colas tan largas* del polinomio de la interpolación. Podría ser buena idea reducir la longitud de la cola.

Buscando Métodos de Simulación para Sistemas Rígidos III

Una razón por la cual no funcionó muy bien la extrapolación era a causa de las *colas tan largas* del polinomio de la interpolación. Podría ser buena idea reducir la longitud de la cola.

Diseñamos entonces un *algoritmo de orden seis* que pasa por tres valores del estado y por cuatro derivadas. De esta manera podemos reducir la longitud de la cola a dos puntos en el pasado:

$$\begin{pmatrix} h \cdot f_{k+1} \\ x_k \\ h \cdot f_k \\ x_{k-1} \\ h \cdot f_{k-1} \\ x_{k-2} \\ h \cdot f_{k-2} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 0 & 1 & -2 & 3 & -4 & 5 & -6 \\ 1 & -2 & 4 & -8 & 16 & -32 & 64 \\ 0 & 1 & -4 & 12 & -32 & 80 & -192 \end{pmatrix} \cdot \mathbf{a}$$

Buscando Métodos de Simulación para Sistemas Rígidos III

Una razón por la cual no funcionó muy bien la extrapolación era a causa de las *colas tan largas* del polinomio de la interpolación. Podría ser buena idea reducir la longitud de la cola.

Diseñamos entonces un *algoritmo de orden seis* que pasa por tres valores del estado y por cuatro derivadas. De esta manera podemos reducir la longitud de la cola a dos puntos en el pasado:

$$\begin{pmatrix} h \cdot f_{k+1} \\ x_k \\ h \cdot f_k \\ x_{k-1} \\ h \cdot f_{k-1} \\ x_{k-2} \\ h \cdot f_{k-2} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 0 & 1 & -2 & 3 & -4 & 5 & -6 \\ 1 & -2 & 4 & -8 & 16 & -32 & 64 \\ 0 & 1 & -4 & 12 & -32 & 80 & -192 \end{pmatrix} \cdot \mathbf{a}$$

Obtenemos un *algoritmo de sexto orden* muy bonito y totalmente desconocido:

$$\mathbf{x_{k+1}} = \frac{3}{11} h \cdot \mathbf{f_{k+1}} - \frac{27}{11} \mathbf{x_k} + \frac{27}{11} h \cdot \mathbf{f_k} + \frac{27}{11} \mathbf{x_{k-1}} + \frac{27}{11} h \cdot \mathbf{f_{k-1}} + \mathbf{x_{k-2}} + \frac{3}{11} h \cdot \mathbf{f_{k-2}}$$

Buscando Métodos de Simulación para Sistemas Rígidos III

Una razón por la cual no funcionó muy bien la extrapolación era a causa de las *colas tan largas* del polinomio de la interpolación. Podría ser buena idea reducir la longitud de la cola.

Diseñamos entonces un *algoritmo de orden seis* que pasa por tres valores del estado y por cuatro derivadas. De esta manera podemos reducir la longitud de la cola a dos puntos en el pasado:

$$\begin{pmatrix} h \cdot f_{k+1} \\ x_k \\ h \cdot f_k \\ x_{k-1} \\ h \cdot f_{k-1} \\ x_{k-2} \\ h \cdot f_{k-2} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 0 & 1 & -2 & 3 & -4 & 5 & -6 \\ 1 & -2 & 4 & -8 & 16 & -32 & 64 \\ 0 & 1 & -4 & 12 & -32 & 80 & -192 \end{pmatrix} \cdot \mathbf{a}$$

Obtenemos un *algoritmo de sexto orden* muy bonito y totalmente desconocido:

$$\mathbf{x_{k+1}} = \frac{3}{11} h \cdot \mathbf{f_{k+1}} - \frac{27}{11} \mathbf{x_k} + \frac{27}{11} h \cdot \mathbf{f_k} + \frac{27}{11} \mathbf{x_{k-1}} + \frac{27}{11} h \cdot \mathbf{f_{k-1}} + \mathbf{x_{k-2}} + \frac{3}{11} h \cdot \mathbf{f_{k-2}}$$

Desafortunadamente ese algoritmo es totalmente inestable.

Buscando Métodos de Simulación para Sistemas Rígidos IV

Tenemos que extender la búsqueda. Ahora permití el uso de valores del estado y de sus derivadas hasta $t = t_{k-5}$, es decir, hasta $s = -5$. Solamente nos interesan métodos implícitos, entonces todos deben incluir la derivada $\mathbf{f_{k+1}}$.

Buscando Métodos de Simulación para Sistemas Rígidos IV

Tenemos que extender la búsqueda. Ahora permití el uso de valores del estado y de sus derivadas hasta $t = t_{k-5}$, es decir, hasta $s = -5$. Solamente nos interesan métodos implícitos, entonces todos deben incluir la derivada $\mathbf{f_{k+1}}$.

Tenemos que escoger 6 elementos de 12, entonces existen 924 métodos diferentes.

Buscando Métodos de Simulación para Sistemas Rígidos IV

Tenemos que extender la búsqueda. Ahora permití el uso de valores del estado y de sus derivadas hasta $t = t_{k-5}$, es decir, hasta $s = -5$. Solamente nos interesan métodos implícitos, entonces todos deben incluir la derivada $\mathbf{f_{k+1}}$.

Tenemos que escoger 6 elementos de 12, entonces existen 924 métodos diferentes.

Programé la búsqueda en Matlab y excluí inmediatamente todos los métodos inestables (la gran mayoría). También excluí todos los métodos estables con dominios de la estabilidad numérica que se cierran en la parte izquierda del plano complejo.

Buscando Métodos de Simulación para Sistemas Rígidos V

Quedaron solamente seis métodos candidatos:

$$(a)\quad \mathbf{x_{k+1}} = \frac{20}{49} h \cdot \mathbf{f_{k+1}} + \frac{120}{49}\mathbf{x_k} - \frac{150}{49}\mathbf{x_{k-1}} + \frac{400}{147}\mathbf{x_{k-2}} - \frac{75}{49}\mathbf{x_{k-3}} + \frac{24}{49}\mathbf{x_{k-4}} - \frac{10}{147}\mathbf{x_{k-5}}$$

$$(b)\quad \mathbf{x_{k+1}} = \frac{308}{745} h \cdot \mathbf{f_{k+1}} + \frac{1776}{745}\mathbf{x_k} - \frac{414}{149}\mathbf{x_{k-1}} + \frac{944}{447}\mathbf{x_{k-2}} - \frac{87}{149}\mathbf{x_{k-3}} - \frac{288}{745} h \cdot \mathbf{f_{k-4}} - \frac{2}{15}\mathbf{x_{k-5}}$$

$$(c)\quad \mathbf{x_{k+1}} = \frac{8820}{21509} h \cdot \mathbf{f_{k+1}} + \frac{52200}{21509}\mathbf{x_k} - \frac{63900}{21509}\mathbf{x_{k-1}} + \frac{400}{157}\mathbf{x_{k-2}} - \frac{28575}{21509}\mathbf{x_{k-3}} + \frac{6984}{21509}\mathbf{x_{k-4}} + \frac{600}{21509} h \cdot \mathbf{f_{k-5}}$$

$$(d)\quad \mathbf{x_{k+1}} = \frac{179028}{432845} h \cdot \mathbf{f_{k+1}} + \frac{206352}{86569}\mathbf{x_k} - \frac{34452}{12367}\mathbf{x_{k-1}} + \frac{26704}{12367}\mathbf{x_{k-2}} - \frac{65547}{86569}\mathbf{x_{k-3}} - \frac{83808}{432845} h \cdot \mathbf{f_{k-4}} + \frac{24}{581} h \cdot \mathbf{f_{k-5}}$$

$$(e)\quad \mathbf{x_{k+1}} = \frac{12}{29} h \cdot \mathbf{f_{k+1}} + \frac{1728}{725}\mathbf{x_k} - \frac{81}{29}\mathbf{x_{k-1}} + \frac{64}{29}\mathbf{x_{k-2}} - \frac{27}{29}\mathbf{x_{k-3}} + \frac{97}{725}\mathbf{x_{k-5}} + \frac{12}{145} h \cdot \mathbf{f_{k-5}}$$

$$(f)\quad \mathbf{x_{k+1}} = \frac{30}{71} h \cdot \mathbf{f_{k+1}} + \frac{162}{71}\mathbf{x_k} - \frac{675}{284}\mathbf{x_{k-1}} + \frac{100}{71}\mathbf{x_{k-2}} - \frac{54}{71}\mathbf{x_{k-4}} + \frac{127}{284}\mathbf{x_{k-5}} + \frac{15}{71} h \cdot \mathbf{f_{k-5}}$$

Entre ellos ya conocemos al método (a). Se trata del *método BDF6*.

Los Dominios de la Estabilidad Numérica

Dos de los seis métodos no sirven porque tienen problemas de estabilidad. Dibujé los dominios de los demás métodos.

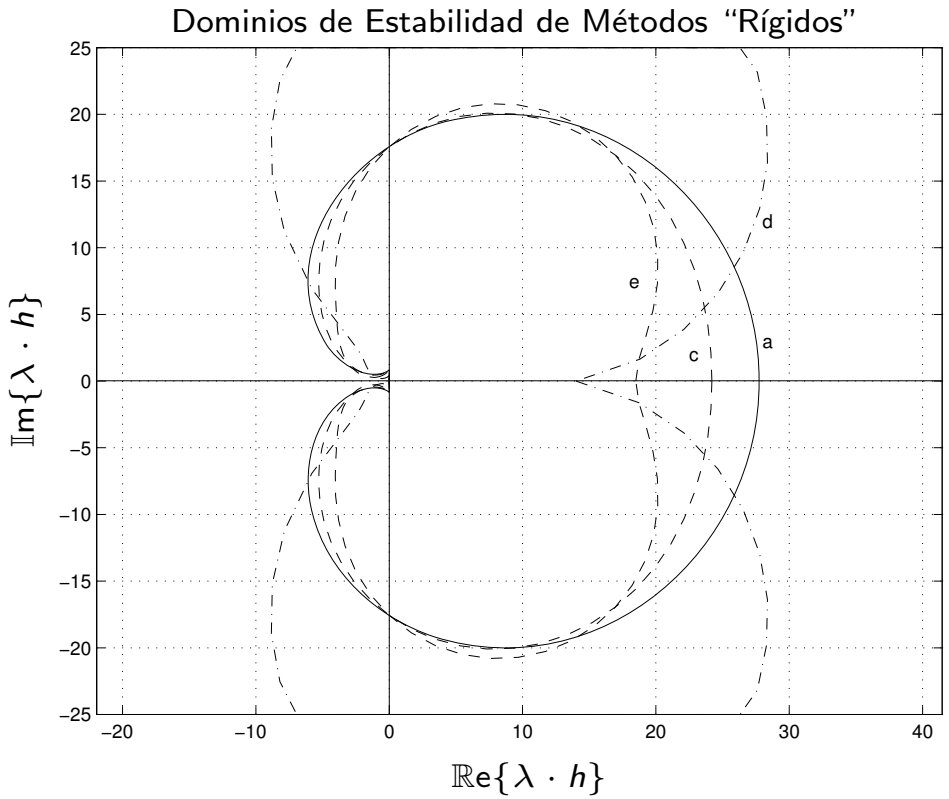

Figure: Dominios de la estabilidad de los cuatro algoritmos "rígidos"

Comparación de los Algoritmos

¿Cómo podemos decidir cuál de esos cuatro algoritmos es el mejor?

Comparación de los Algoritmos

¿Cómo podemos decidir cuál de esos cuatro algoritmos es el mejor?

Por un lado podemos analizar el "daño" que ocurre a causa de la región inestable a la izquierda del eje imaginario.

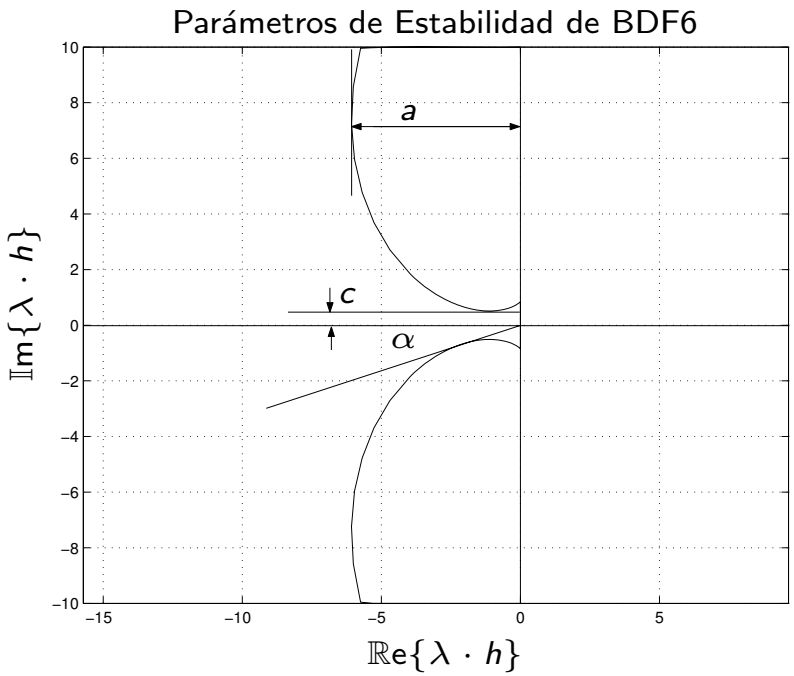

Figure: Los parámetros de la estabilidad de BDF6

Comparación de los Algoritmos II

Comparación de los Algoritmos II

- Es posible definir un ángulo α con el eje real negativo que define un *área totalmente estable*.

Comparación de los Algoritmos II

- Es posible definir un ángulo α con el eje real negativo que define un *área totalmente estable*.
- Algoritmos que no son A-estables pueden aun ser caracterizados como *A(α)-estables*. El algoritmo BDF6 es A(α)-estable con un ángulo de $\alpha = 19^{o}$.

Comparación de los Algoritmos II

- Es posible definir un ángulo α con el eje real negativo que define un *área totalmente estable*.
- Algoritmos que no son A-estables pueden aun ser caracterizados como *A(α)-estables*. El algoritmo BDF6 es A(α)-estable con un ángulo de $\alpha = 19^o$.
- También es posible definir las distancias *a* del eje real negativo y *c* del eje imaginario que excluyen todas las regiones inestables.

Comparación de los Algoritmos II

- Es posible definir un ángulo α con el eje real negativo que define un *área totalmente estable*.
- Algoritmos que no son A-estables pueden aun ser caracterizados como *A(α)-estables*. El algoritmo BDF6 es A(α)-estable con un ángulo de $\alpha = 19^o$.
- También es posible definir las distancias *a* del eje real negativo y *c* del eje imaginario que excluyen todas las regiones inestables.
- Los tres parámetros juntos caracterizan bien el "daño" que ocurre a causa de la región inestable dentro de la parte izquierda del plano complejo.

Comparación de los Algoritmos III

Por otro lado también tenemos que tener en cuenta la *precisión de los algoritmos*.
Todos los algoritmos lineales en múltiples pasos pueden escribirse de la forma siguiente:

$$\sum_{i=0}^{m} \alpha_i \cdot \mathbf{x_{k+i}} + h \cdot \sum_{i=0}^{m} \beta_i \cdot \mathbf{f_{k+i}} = 0$$

Es posible obtener una fórmula para el *coeficiente del error*:

$$c_{err} = \sum_{i=0}^{m} \left(\frac{1}{(n+1)!} \cdot i^{n+1} \cdot \alpha_i - \frac{1}{n!} \cdot i^{n} \cdot \beta_i \right)$$

Comparación de los Algoritmos III

Por otro lado también tenemos que tener en cuenta la *precisión de los algoritmos*. Todos los algoritmos lineales en múltiples pasos pueden escribirse de la forma siguiente:

$$\sum_{i=0}^{m} \alpha_i \cdot \mathbf{x_{k+i}} + h \cdot \sum_{i=0}^{m} \beta_i \cdot \mathbf{f_{k+i}} = 0$$

Es posible obtener una fórmula para el *coeficiente del error*:

$$c_{err} = \sum_{i=0}^{m} \left(\frac{1}{(n+1)!} \cdot i^{n+1} \cdot \alpha_i - \frac{1}{n!} \cdot i^{n} \cdot \beta_i \right)$$

Comparando los coeficientes del error de los métodos BDF con los de los métodos Adams, se nota que los coeficientes del error de los métodos Adams son bastante más pequeños.

Comparación de los Algoritmos III

Por otro lado también tenemos que tener en cuenta la *precisión de los algoritmos*. Todos los algoritmos lineales en múltiples pasos pueden escribirse de la forma siguiente:

$$\sum_{i=0}^{m} \alpha_i \cdot \mathbf{x_{k+i}} + h \cdot \sum_{i=0}^{m} \beta_i \cdot \mathbf{f_{k+i}} = 0$$

Es posible obtener una fórmula para el *coeficiente del error*:

$$c_{err} = \sum_{i=0}^{m} \left(\frac{1}{(n+1)!} \cdot i^{n+1} \cdot \alpha_i - \frac{1}{n!} \cdot i^{n} \cdot \beta_i \right)$$

Comparando los coeficientes del error de los métodos BDF con los de los métodos Adams, se nota que los coeficientes del error de los métodos Adams son bastante más pequeños.

Por eso, si estamos usando un algoritmo en múltiples pasos para la *simulación de un sistema no rígido*, obtenemos mejor precisión para el mismo tamaño del paso h (es decir, para el mismo "precio") con Adams-Moulton que con BDF.

Comparación de los Algoritmos III

Por otro lado también tenemos que tener en cuenta la *precisión de los algoritmos*. Todos los algoritmos lineales en múltiples pasos pueden escribirse de la forma siguiente:

$$\sum_{i=0}^{m} \alpha_i \cdot \mathbf{x_{k+i}} + h \cdot \sum_{i=0}^{m} \beta_i \cdot \mathbf{f_{k+i}} = 0$$

Es posible obtener una fórmula para el *coeficiente del error*:

$$c_{err} = \sum_{i=0}^{m} \left(\frac{1}{(n+1)!} \cdot i^{n+1} \cdot \alpha_i - \frac{1}{n!} \cdot i^{n} \cdot \beta_i \right)$$

Comparando los coeficientes del error de los métodos BDF con los de los métodos Adams, se nota que los coeficientes del error de los métodos Adams son bastante más pequeños.

Por eso, si estamos usando un algoritmo en múltiples pasos para la *simulación de un sistema no rígido*, obtenemos mejor precisión para el mismo tamaño del paso h (es decir, para el mismo "precio") con Adams-Moulton que con BDF.

Entonces, será útil tener en cuenta el tamaño del coeficiente del error en el análisis de la calidad de un algoritmo lineal en múltiples pasos.

Comparación de los Algoritmos IV

Será útil también mirar el *gráfico de amortiguación* de cada algoritmo.

Comparación de los Algoritmos IV

Será útil también mirar el *gráfico de amortiguación* de cada algoritmo.

En el análisis del comportamiento de la amortiguación de los algoritmos en un solo paso se usó el sistema escalar. El factor de la amortiguación se definió como:

$$\hat{\sigma}_d = -\log(\mathrm{abs}(f))$$

Comparación de los Algoritmos IV

Será útil también mirar el *gráfico de amortiguación* de cada algoritmo.

En el análisis del comportamiento de la amortiguación de los algoritmos en un solo paso se usó el sistema escalar. El factor de la amortiguación se definió como:

$$\hat{\sigma}_d = -\log(\mathrm{abs}(f))$$

En el análisis del comportamiento de la amortiguación de los algoritmos en múltiples pasos esa definición ya no sirve más, porque aun un sistema continuo escalar tiene una matriz **F**.

Comparación de los Algoritmos IV

Será útil también mirar el *gráfico de amortiguación* de cada algoritmo.

En el análisis del comportamiento de la amortiguación de los algoritmos en un solo paso se usó el sistema escalar. El factor de la amortiguación se definió como:

$$\hat{\sigma}_d = -\log(\mathrm{abs}(f))$$

En el análisis del comportamiento de la amortiguación de los algoritmos en múltiples pasos esa definición ya no sirve más, porque aun un sistema continuo escalar tiene una matriz **F**.

Entonces tendremos que extender la definición:

$$\hat{\sigma}_d = -\log(\max(\mathrm{abs}(\mathrm{eig}(\mathbf{F}))))$$

Comparación de los Algoritmos IV

Será útil también mirar el *gráfico de amortiguación* de cada algoritmo.

En el análisis del comportamiento de la amortiguación de los algoritmos en un solo paso se usó el sistema escalar. El factor de la amortiguación se definió como:

$$\hat{\sigma}_d = -\log(\mathrm{abs}(f))$$

En el análisis del comportamiento de la amortiguación de los algoritmos en múltiples pasos esa definición ya no sirve más, porque aun un sistema continuo escalar tiene una matriz **F**.

Entonces tendremos que extender la definición:

$$\hat{\sigma}_d = -\log(\max(\mathrm{abs}(\mathrm{eig}(\mathbf{F}))))$$

Dibujaremos ahora no solamente el gráfico de amortiguación lineal, sino también el gráfico de amortiguación logarítmico.

El Gráfico de Amortiguación de los Algoritmos

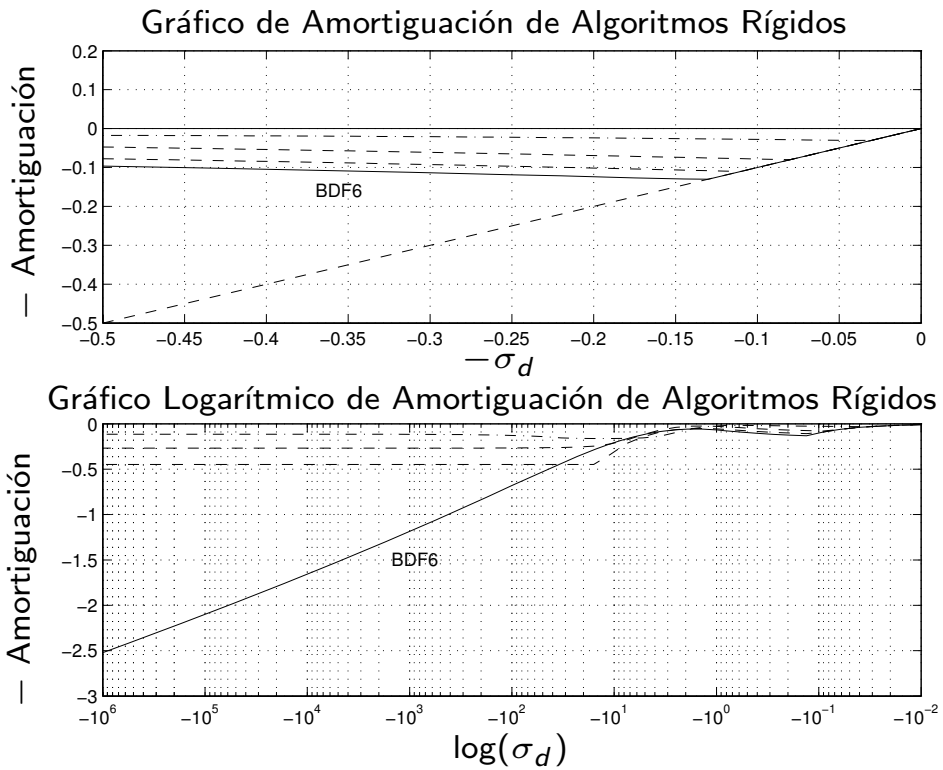

Figure: Gráfico de amortiguación de los algoritmos rígidos

El Gráfico de Amortiguación de los Algoritmos

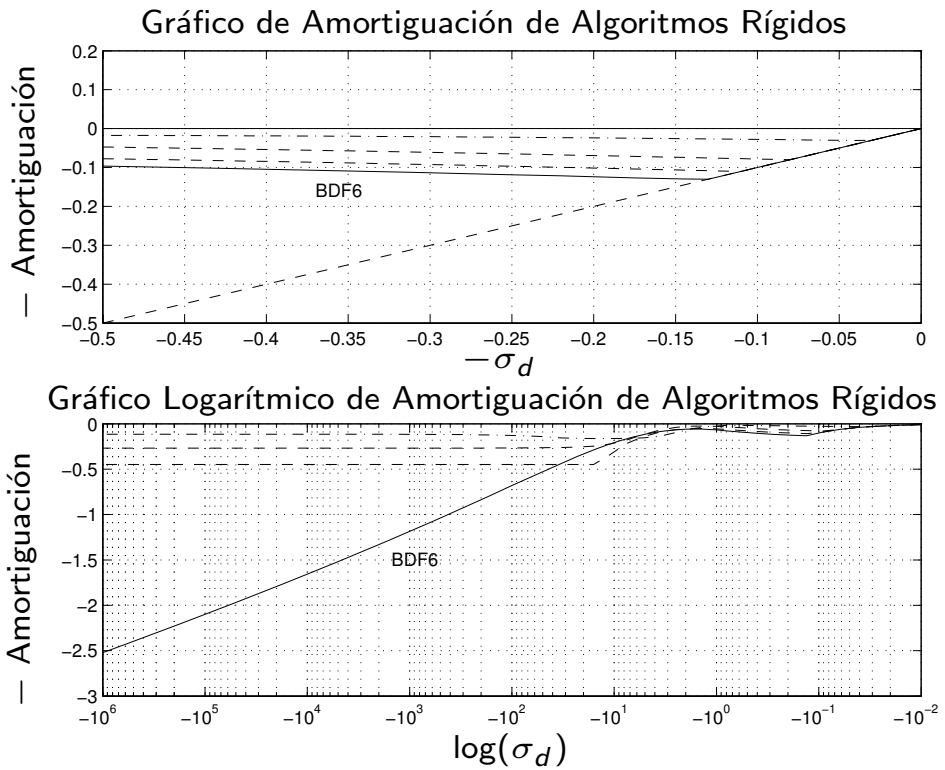

Figure: Gráfico de amortiguación de los algoritmos rígidos

Se nota que solamente el algoritmo BDF es L(α)-estable. Los demás algoritmos no lo son.

Comparación de los Algoritmos V

Comparación de los Algoritmos V

- BDF6 aparentemente es el mejor de los algoritmos de orden seis en múltiples pasos para la simulación de sistemas rígidos.

Comparación de los Algoritmos V

- BDF6 aparentemente es el mejor de los algoritmos de orden seis en múltiples pasos para la simulación de sistemas rígidos.
- Todos los algoritmos que sobrevivieron la búsqueda explotan el rango de valores hasta t_{k-5}. Aparentemente fue muy mala idea tratar de cortar la longitud de la cola.

Comparación de los Algoritmos V

- BDF6 aparentemente es el mejor de los algoritmos de orden seis en múltiples pasos para la simulación de sistemas rígidos.
- Todos los algoritmos que sobrevivieron la búsqueda explotan el rango de valores hasta t_{k-5}. Aparentemente fue muy mala idea tratar de cortar la longitud de la cola.
- Ninguno de los algoritmos que sobrevivió la búsqueda usa más de un valor de la derivada del pasado. Aparentemente es mala idea usar derivadas del pasado en el desarrollo de algoritmos para la simulación de sistemas rígidos.

Extender la Búsqueda

Extender la Búsqueda

- Posiblemente sería buena idea alargar la cola. Como ya decidimos que las derivadas del pasado no sirven, limitamos la búsqueda de tal manera que solamente consideramos algoritmos con valores del estado. Extendemos la cola hasta t_{k-11}.

Extender la Búsqueda

- Posiblemente sería buena idea alargar la cola. Como ya decidimos que las derivadas del pasado no sirven, limitamos la búsqueda de tal manera que solamente consideramos algoritmos con valores del estado. Extendemos la cola hasta t_{k-11}.
- De nuevo existen 924 posibles algoritmos de orden seis. Entre ellos, 314 tienes características similares a las del algoritmo BDF6.

Extender la Búsqueda

- Posiblemente sería buena idea alargar la cola. Como ya decidimos que las derivadas del pasado no sirven, limitamos la búsqueda de tal manera que solamente consideramos algoritmos con valores del estado. Extendemos la cola hasta t_{k-11}.
- De nuevo existen 924 posibles algoritmos de orden seis. Entre ellos, 314 tienes características similares a las del algoritmo BDF6.

Sus cinco parámetros de desempeño son:

BDF6	Demás métodos rígidos
$\alpha = 19^{\circ}$	$\alpha \in [19^{\circ}, 48^{\circ}]$
$a = -6.0736$	$a \in [-6.0736, -0.6619]$
$c = 0.5107$	$c \in [0.2250, 0.8316]$
$c_{err} = -0.0583$	$c_{err} \in [-7.4636, -0.0583]$
$reg.as. = -0.14$	$reg.as. \in [-0.30, -0.01]$

Table: Propiedades de los algoritmos rígidos de orden seis.

Extender la Búsqueda

- Posiblemente sería buena idea alargar la cola. Como ya decidimos que las derivadas del pasado no sirven, limitamos la búsqueda de tal manera que solamente consideramos algoritmos con valores del estado. Extendemos la cola hasta t_{k-11}.
- De nuevo existen 924 posibles algoritmos de orden seis. Entre ellos, 314 tienes características similares a las del algoritmo BDF6.

Sus cinco parámetros de desempeño son:

BDF6	Demás métodos rígidos
$\alpha = 19^{\circ}$	$\alpha \in [19^{\circ}, 48^{\circ}]$
$a = -6.0736$	$a \in [-6.0736, -0.6619]$
$c = 0.5107$	$c \in [0.2250, 0.8316]$
$c_{err} = -0.0583$	$c_{err} \in [-7.4636, -0.0583]$
$reg.as. = -0.14$	$reg.as. \in [-0.30, -0.01]$

Table: Propiedades de los algoritmos rígidos de orden seis.

Para evaluar los métodos de forma cuantitativa necesitamos un índice de desempeño:

$$P.I._i = \frac{|\alpha_i|}{\|\alpha\|} - \frac{|a_i|}{\|a\|} + \frac{|c_i|}{\|c\|} - k \cdot \frac{|c_{err_i}|}{\|c_{err}\|} + \frac{|as.reg._i|}{\|as.reg.\|} = max!$$

Usamos $k = 20$ para indicar que un pequeño coeficiente del error es muy importante.

Extender la Búsqueda II

Los tres mejores algoritmos son:

$$\mathbf{x_{k+1}} = \frac{72}{167} h \cdot \mathbf{f_{k+1}} + \frac{2592}{1169}\mathbf{x_k} - \frac{2592}{1169}\mathbf{x_{k-1}} + \frac{1152}{835}\mathbf{x_{k-2}} \tag{1}$$

$$- \frac{324}{835}\mathbf{x_{k-3}} + \frac{81}{5845}\mathbf{x_{k-7}} - \frac{32}{5845}\mathbf{x_{k-8}} \tag{2}$$

$$\mathbf{x_{k+1}} = \frac{420}{977} h \cdot \mathbf{f_{k+1}} + \frac{19600}{8793}\mathbf{x_k} - \frac{2205}{977}\mathbf{x_{k-1}} + \frac{1400}{977}\mathbf{x_{k-2}} \tag{3}$$

$$- \frac{1225}{2931}\mathbf{x_{k-3}} + \frac{40}{2931}\mathbf{x_{k-6}} - \frac{7}{8793}\mathbf{x_{k-9}} \tag{4}$$

$$\mathbf{x_{k+1}} = \frac{44}{103} h \cdot \mathbf{f_{k+1}} + \frac{5808}{2575}\mathbf{x_k} - \frac{242}{103}\mathbf{x_{k-1}} + \frac{484}{309}\mathbf{x_{k-2}} \tag{5}$$

$$- \frac{363}{721}\mathbf{x_{k-3}} + \frac{242}{7725}\mathbf{x_{k-5}} - \frac{4}{18025}\mathbf{x_{k-10}} \tag{6}$$

Sus parámetros de desempeño son:

BDF6	SS6a	SS6b	SS6c
$\alpha = 19^\circ$	$\alpha = 45^\circ$	$\alpha = 44^\circ$	$\alpha = 43^\circ$
$a = -6.0736$	$a = -2.6095$	$a = -2.7700$	$a = -3.0839$
$c = 0.5107$	$c = 0.7994$	$c = 0.8048$	$c = 0.8156$
$c_{err} = -0.0583$	$c_{err} = -0.1478$	$c_{err} = -0.1433$	$c_{err} = -0.1343$
$reg.as. = -0.14$	$reg.as. = -0.21$	$reg.as. = -0.21$	$reg.as. = -0.21$

Table: Propiedades de los algoritmos rígidos de orden seis.

Extender la Búsqueda III

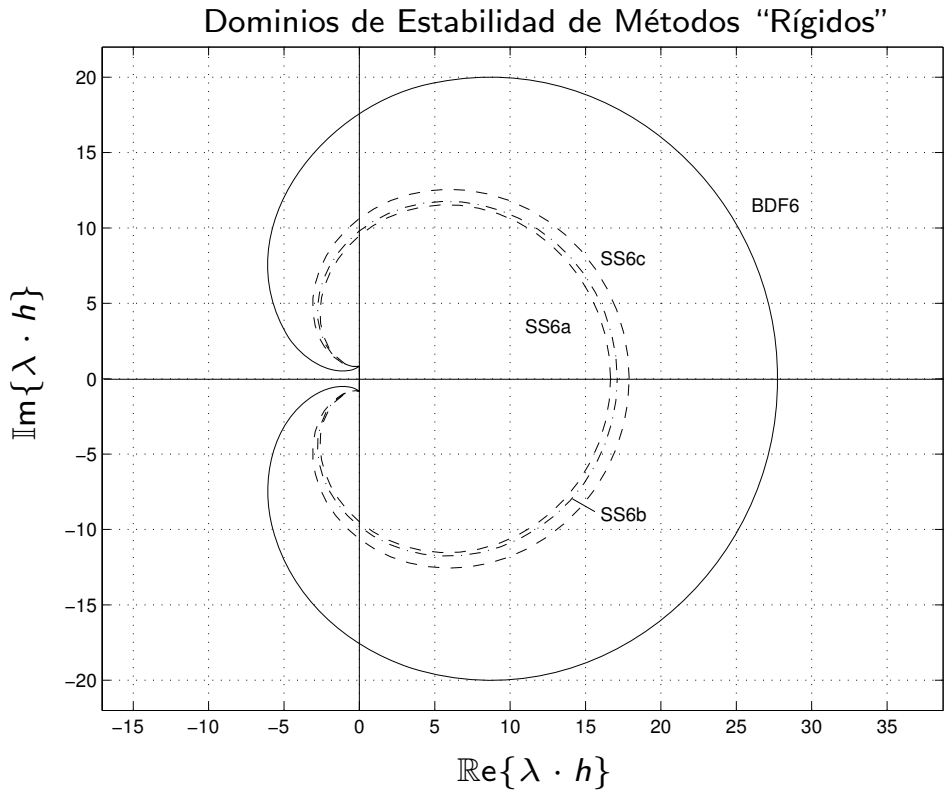

Figure: Dominios de estabilidad de los tres algoritmos "rígidos" y de BDF6

Extender la Búsqueda IV

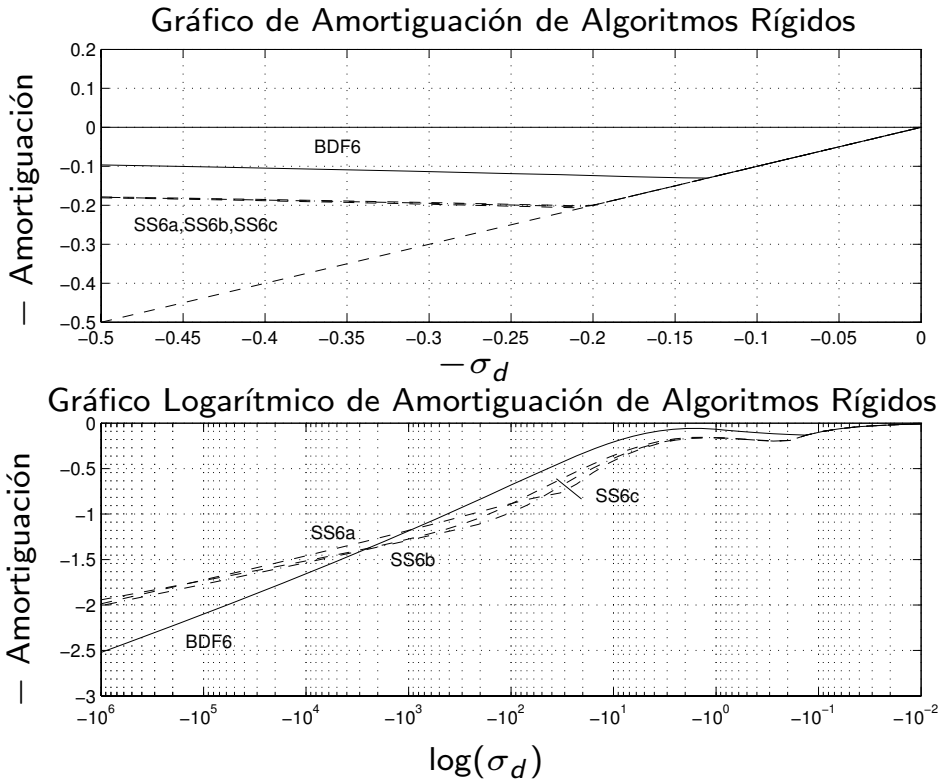

Figure: Gráfico de amortiguación de los algoritmos rígidos

Extender la Búsqueda V

Extender la Búsqueda V

- **Por fin encontramos unos métodos de orden seis en múltiples pasos mejores que BDF6.**

Extender la Búsqueda V

- **Por fin encontramos unos métodos de orden seis en múltiples pasos mejores que BDF6.**
- Los nuevos algoritmos pueden usarse también para la *simulación de sistemas con oscilaciones*, porque su ángulo α es bastante más grande que en el caso de BDF6.

Extender la Búsqueda V

- **Por fin encontramos unos métodos de orden seis en múltiples pasos mejores que BDF6.**
- Los nuevos algoritmos pueden usarse también para la *simulación de sistemas con oscilaciones*, porque su ángulo α es bastante más grande que en el caso de BDF6.
- La característica más importante de esos tres algoritmos es su *región asintótica* que es casi 50% más grande que en el caso de BDF6. Eso permitirá usar pasos más grandes durante la simulación.

Extender la Búsqueda V

- **Por fin encontramos unos métodos de orden seis en múltiples pasos mejores que BDF6.**
- Los nuevos algoritmos pueden usarse también para la *simulación de sistemas con oscilaciones*, porque su ángulo α es bastante más grande que en el caso de BDF6.
- La característica más importante de esos tres algoritmos es su *región asintótica* que es casi 50% más grande que en el caso de BDF6. Eso permitirá usar pasos más grandes durante la simulación.
- El *coeficiente del error*, c_{err}, de esos nuevos métodos es un poco más grande que en el caso de BDF6, pero eso no importa. El paso de integración en BDF6 es mucho más frecuentemente limitado por la región asintótica que por el tamaño de c_{err}.

Fórmulas de Diferencia hacia Atrás de Alto Orden

Fórmulas de Diferencia hacia Atrás de Alto Orden

- Usando la misma metodología pudimos encontrar *fórmulas de diferencia hacia atrás de órdenes 7..9*.

Fórmulas de Diferencia hacia Atrás de Alto Orden

- Usando la misma metodología pudimos encontrar *fórmulas de diferencia hacia atrás de órdenes 7..9*.
- Desafortunadamente, estos métodos tienen *ángulos α más pequeños* de nuevo. Para los órdenes siete y ocho todavía pueden encontrarse métodos con $\alpha = 35^{o}$. Para el orden nueve el ángulo más grande es de $\alpha = 18^{o}$.

Fórmulas de Diferencia hacia Atrás de Alto Orden

- Usando la misma metodología pudimos encontrar *fórmulas de diferencia hacia atrás de órdenes 7..9*.
- Desafortunadamente, estos métodos tienen *ángulos α más pequeños* de nuevo. Para los órdenes siete y ocho todavía pueden encontrarse métodos con $\alpha = 35^{o}$. Para el orden nueve el ángulo más grande es de $\alpha = 18^{o}$.
- También la *región asintótica* va disminuyendo. Para los órdenes siete y ocho todavía pueden encontrarse métodos con $reg.as. = -0.14$. Para el orden nueve la región más grande es de $reg.as. = -0.10$.

Fórmulas de Diferencia hacia Atrás de Alto Orden

- Usando la misma metodología pudimos encontrar *fórmulas de diferencia hacia atrás de órdenes 7..9*.
- Desafortunadamente, estos métodos tienen *ángulos α más pequeños* de nuevo. Para los órdenes siete y ocho todavía pueden encontrarse métodos con $\alpha = 35^o$. Para el orden nueve el ángulo más grande es de $\alpha = 18^o$.
- También la *región asintótica* va disminuyendo. Para los órdenes siete y ocho todavía pueden encontrarse métodos con $reg.as. = -0.14$. Para el orden nueve la región más grande es de $reg.as. = -0.10$.
- El *coeficiente del error*, c_{err}, también crece rápidamente. Sin embargo no tiene sentido comparar coeficientes del error entre métodos de diferentes órdenes, solamente entre métodos del mismo orden.

Fórmulas de Diferencia hacia Atrás de Alto Orden

- Usando la misma metodología pudimos encontrar *fórmulas de diferencia hacia atrás de órdenes 7..9*.
- Desafortunadamente, estos métodos tienen *ángulos α más pequeños* de nuevo. Para los órdenes siete y ocho todavía pueden encontrarse métodos con $\alpha = 35^o$. Para el orden nueve el ángulo más grande es de $\alpha = 18^o$.
- También la *región asintótica* va disminuyendo. Para los órdenes siete y ocho todavía pueden encontrarse métodos con $reg.as. = -0.14$. Para el orden nueve la región más grande es de $reg.as. = -0.10$.
- El *coeficiente del error*, c_{err}, también crece rápidamente. Sin embargo no tiene sentido comparar coeficientes del error entre métodos de diferentes órdenes, solamente entre métodos del mismo orden.
- Métodos de órdenes tan altos son útiles para la simulación de *problemas de la mecánica celestial*. Sin embargo, en los métodos de órdenes tan altos el *error por redondeo* en doble precisión ya es más grande que el *error por truncamiento*. Estos algoritmos deben implementarse entonces en *precisión cuádruple*.

La Iteración de Newton

Ya sabemos que los algoritmos del tipo BDF son implícitos. Entonces tenemos que iterar en cada paso.

Podemos escribir los métodos BDF de la forma:

$$\mathbf{x}_{k+1} = \alpha_i \; h \cdot \mathbf{f}_{k+1} + \sum_{j=1}^{i} \beta_{ij} \; \mathbf{x}_{k-j+1}$$

Moviendo todo al lado derecho:

$$\mathcal{F}(\mathbf{x}_{k+1}) = \alpha_i \; h \cdot \mathbf{f}(\mathbf{x}_{k+1}, t_{k+1}) - \mathbf{x}_{k+1} + \sum_{j=1}^{i} \beta_{ij} \; \mathbf{x}_{k-j+1} = 0.0$$

La Iteración de Newton

Ya sabemos que los algoritmos del tipo BDF son implícitos. Entonces tenemos que iterar en cada paso.

Podemos escribir los métodos BDF de la forma:

$$\mathbf{x}_{k+1} = \alpha_i \; h \cdot \mathbf{f}_{k+1} + \sum_{j=1}^{i} \beta_{ij} \; \mathbf{x}_{k-j+1}$$

Moviendo todo al lado derecho:

$$\mathcal{F}(\mathbf{x}_{k+1}) = \alpha_i \; h \cdot \mathbf{f}(\mathbf{x}_{k+1}, t_{k+1}) - \mathbf{x}_{k+1} + \sum_{j=1}^{i} \beta_{ij} \; \mathbf{x}_{k-j+1} = 0.0$$

Ahora se puede usar la iteración de Newton:

$$\mathbf{x}_{k+1}^{\ell+1} = \mathbf{x}_{k+1}^{\ell} - [\mathcal{H}^{\ell}]^{-1} \cdot [\mathcal{F}^{\ell}]$$

donde $\mathcal{H}$ es el *Hessiano de la función vectorial*, $\mathcal{F}$:

$$\mathcal{H} = \alpha_i \cdot (\mathcal{J} \cdot h) - \mathbf{I}^{(\mathbf{n})}$$

y $\mathcal{J}$ es el *Jacobiano del sistema*.

La Economía de los Algoritmos AB

Simulamos un sistema lineal de quinto orden usando diferentes algoritmos AB con diferentes pasos fijos. Comparamos los errores globales de la simulación. También comparamos con el algoritmo RK4.

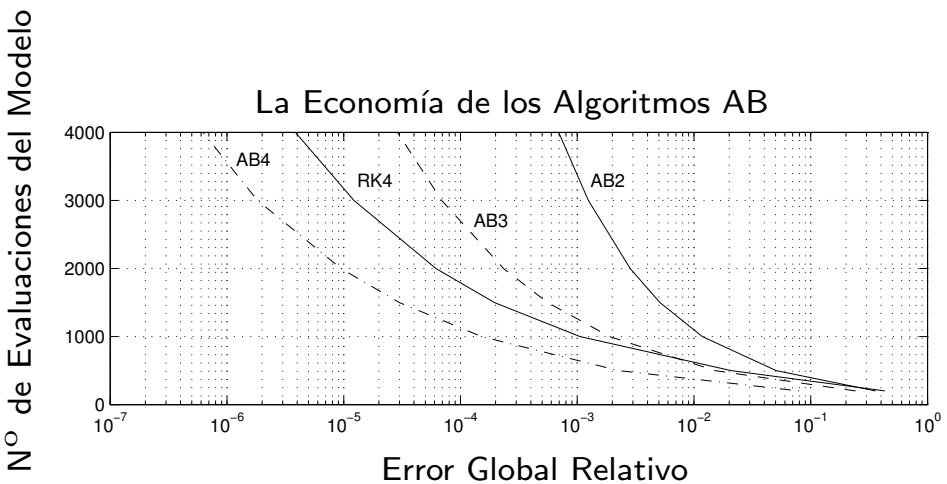

Figure: La economía de los algoritmos AB

La Economía de los Algoritmos AB II

La Economía de los Algoritmos AB II

Nos damos cuenta que:

- Para el mismo número de evaluaciones del modelo (es decir, para el mismo "precio" de la simulación) obtenemos apróximadamente una precisión mejorada por un factor de 10 si aumentamos el orden del algoritmo por uno.

La Economía de los Algoritmos AB II

Nos damos cuenta que:

- Para el mismo número de evaluaciones del modelo (es decir, para el mismo "precio" de la simulación) obtenemos apróximadamente una precisión mejorada por un factor de 10 si aumentamos el orden del algoritmo por uno.
- Usando un método del orden n podemos obtener económicamente una precisión de 10^{-n}.

La Economía de los Algoritmos AB II

Nos damos cuenta que:

- Para el mismo número de evaluaciones del modelo (es decir, para el mismo "precio" de la simulación) obtenemos apróximadamente una precisión mejorada por un factor de 10 si aumentamos el orden del algoritmo por uno.
- Usando un método del orden n podemos obtener económicamente una precisión de 10^{-n}.
- En ese ejemplo, AB4 es más económico que RK4 porque se trata de un sistema lineal sin discontinuidades. Sin embargo, si añadimos el control del paso, la situación cambiará, porque el control del paso es más económico para algoritmos en un solo paso que para algoritmos en múltiples pasos.

El Control del Paso

Los algoritmos lineales en múltiples pasos se diseñaron bajo la suposición de un *muestreo equidistante*. Si *cambiamos la longitud del paso de la integración* entre un paso y el próximo, esa suposición no es más válida.

Si queremos usar los métodos en múltiples pasos con un control del paso tenemos que servirnos de un truco. De nuevo nos ayudarán los polinomios de Newton-Gregory.

El Control del Paso

Los algoritmos lineales en múltiples pasos se diseñaron bajo la suposición de un *muestreo equidistante*. Si *cambiamos la longitud del paso de la integración* entre un paso y el próximo, esa suposición no es más válida.

Si queremos usar los métodos en múltiples pasos con un control del paso tenemos que servirnos de un truco. De nuevo nos ayudarán los polinomios de Newton-Gregory.

Escribimos el polinomio de Newton-Gregory hacia atrás del vector del estado:

$$\mathbf{x}(t) = \mathbf{x_k} + s\nabla\mathbf{x}_k + \left(\frac{s^2}{2} + \frac{s}{2}\right)\nabla^2\mathbf{x}_k + \left(\frac{s^3}{6} + \frac{s^2}{2} + \frac{s}{3}\right)\nabla^3\mathbf{x}_k + \ldots$$

El Control del Paso

Los algoritmos lineales en múltiples pasos se diseñaron bajo la suposición de un *muestreo equidistante*. Si *cambiamos la longitud del paso de la integración* entre un paso y el próximo, esa suposición no es más válida.

Si queremos usar los métodos en múltiples pasos con un control del paso tenemos que servirnos de un truco. De nuevo nos ayudarán los polinomios de Newton-Gregory.

Escribimos el polinomio de Newton-Gregory hacia atrás del vector del estado:

$$\mathbf{x}(t) = \mathbf{x_k} + s\nabla\mathbf{x}_k + \left(\frac{s^2}{2} + \frac{s}{2}\right)\nabla^2\mathbf{x}_k + \left(\frac{s^3}{6} + \frac{s^2}{2} + \frac{s}{3}\right)\nabla^3\mathbf{x}_k + \ldots$$

Diferenciamos con respecto al tiempo:

$$\dot{\mathbf{x}}(t) = \frac{1}{h}\left[\nabla\mathbf{x}_k + \left(s + \frac{1}{2}\right)\nabla^2\mathbf{x}_k + \left(\frac{s^2}{2} + s + \frac{1}{3}\right)\nabla^3\mathbf{x}_k + \ldots\right]$$

El Control del Paso

Los algoritmos lineales en múltiples pasos se diseñaron bajo la suposición de un *muestreo equidistante*. Si *cambiamos la longitud del paso de la integración* entre un paso y el próximo, esa suposición no es más válida.

Si queremos usar los métodos en múltiples pasos con un control del paso tenemos que servirnos de un truco. De nuevo nos ayudarán los polinomios de Newton-Gregory.

Escribimos el polinomio de Newton-Gregory hacia atrás del vector del estado:

$$\mathbf{x}(t) = \mathbf{x_k} + s\nabla\mathbf{x}_k + \left(\frac{s^2}{2} + \frac{s}{2}\right)\nabla^2\mathbf{x}_k + \left(\frac{s^3}{6} + \frac{s^2}{2} + \frac{s}{3}\right)\nabla^3\mathbf{x}_k + \dots$$

Diferenciamos con respecto al tiempo:

$$\dot{\mathbf{x}}(t) = \frac{1}{h}\left[\nabla\mathbf{x}_k + \left(s + \frac{1}{2}\right)\nabla^2\mathbf{x}_k + \left(\frac{s^2}{2} + s + \frac{1}{3}\right)\nabla^3\mathbf{x}_k + \dots\right]$$

La segunda derivada puede escribirse de la forma:

$$\ddot{\mathbf{x}}(t) = \frac{1}{h^2}\left[\nabla^2\mathbf{x}_k + (s+1)\nabla^3\mathbf{x}_k + \dots\right]$$

etc.

El Control del Paso II

Truncando después del término cúbico y evaluando por $t = t_k$, es decir, $s = 0.0$, obtenemos:

$$\begin{pmatrix} x_k \\ h \cdot \dot{x}_k \\ \frac{h^2}{2} \cdot \ddot{x}_k \\ \frac{h^3}{6} \cdot x_k^{(iii)} \end{pmatrix} = \frac{1}{6} \cdot \begin{pmatrix} 6 & 0 & 0 & 0 \\ 11 & -18 & 9 & -2 \\ 6 & -15 & 12 & -3 \\ 1 & -3 & 3 & -1 \end{pmatrix} \cdot \begin{pmatrix} x_k \\ x_{k-1} \\ x_{k-2} \\ x_{k-3} \end{pmatrix}$$

El Control del Paso II

Truncando después del término cúbico y evaluando por $t = t_k$, es decir, $s = 0.0$, obtenemos:

$$\begin{pmatrix} x_k \\ h \cdot \dot{x}_k \\ \frac{h^2}{2} \cdot \ddot{x}_k \\ \frac{h^3}{6} \cdot x_k^{(iii)} \end{pmatrix} = \frac{1}{6} \cdot \begin{pmatrix} 6 & 0 & 0 & 0 \\ 11 & -18 & 9 & -2 \\ 6 & -15 & 12 & -3 \\ 1 & -3 & 3 & -1 \end{pmatrix} \cdot \begin{pmatrix} x_k \\ x_{k-1} \\ x_{k-2} \\ x_{k-3} \end{pmatrix}$$

El vector de la izquierda se llama el *vector de Nordsieck*, aquí de tercer orden.

El Control del Paso II

Truncando después del término cúbico y evaluando por $t = t_k$, es decir, $s = 0.0$, obtenemos:

$$\begin{pmatrix} x_k \\ h \cdot \dot{x}_k \\ \frac{h^2}{2} \cdot \ddot{x}_k \\ \frac{h^3}{6} \cdot x_k^{(iii)} \end{pmatrix} = \frac{1}{6} \cdot \begin{pmatrix} 6 & 0 & 0 & 0 \\ 11 & -18 & 9 & -2 \\ 6 & -15 & 12 & -3 \\ 1 & -3 & 3 & -1 \end{pmatrix} \cdot \begin{pmatrix} x_k \\ x_{k-1} \\ x_{k-2} \\ x_{k-3} \end{pmatrix}$$

El vector de la izquierda se llama el *vector de Nordsieck*, aquí de tercer orden.

Si tenemos n valores equidistantes de una variable del estado, podemos calcular *sin pérdida de precisión* en su lugar el valor de la variable misma más el valor de sus primeras $n-1$ derivadas en el instante t_k.

El Control del Paso III

Ahora es fácil cambiar el paso:

$$\begin{pmatrix} x_k \\ h_{\text{nuevo}} \cdot \dot{x}_k \\ \frac{h_{\text{nuevo}}^2}{2} \cdot \ddot{x}_k \\ \frac{h_{\text{nuevo}}^3}{6} \cdot x_k^{(iii)} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{h_{\text{nuevo}}}{h_{\text{viejo}}} & 0 & 0 \\ 0 & 0 & \left(\frac{h_{\text{nuevo}}}{h_{\text{viejo}}}\right)^2 & 0 \\ 0 & 0 & 0 & \left(\frac{h_{\text{nuevo}}}{h_{\text{viejo}}}\right)^3 \end{pmatrix} \cdot \begin{pmatrix} x_k \\ h_{\text{viejo}} \cdot \dot{x}_k \\ \frac{h_{\text{viejo}}^2}{2} \cdot \ddot{x}_k \\ \frac{h_{\text{viejo}}^3}{6} \cdot x_k^{(iii)} \end{pmatrix}$$

Ahora tenemos el vector de Nordsieck expresado en el nuevo paso h_{nuevo}.

El Control del Paso III

Ahora es fácil cambiar el paso:

$$\begin{pmatrix} x_k \\ h_{\text{nuevo}} \cdot \dot{x}_k \\ \frac{h_{\text{nuevo}}^2}{2} \cdot \ddot{x}_k \\ \frac{h_{\text{nuevo}}^3}{6} \cdot x_k^{(iii)} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{h_{\text{nuevo}}}{h_{\text{viejo}}} & 0 & 0 \\ 0 & 0 & \left(\frac{h_{\text{nuevo}}}{h_{\text{viejo}}}\right)^2 & 0 \\ 0 & 0 & 0 & \left(\frac{h_{\text{nuevo}}}{h_{\text{viejo}}}\right)^3 \end{pmatrix} \cdot \begin{pmatrix} x_k \\ h_{\text{viejo}} \cdot \dot{x}_k \\ \frac{h_{\text{viejo}}^2}{2} \cdot \ddot{x}_k \\ \frac{h_{\text{viejo}}^3}{6} \cdot x_k^{(iii)} \end{pmatrix}$$

Ahora tenemos el vector de Nordsieck expresado en el nuevo paso h_{nuevo}.

En consecuencia se puede escribir:

$$\begin{pmatrix} x_k \\ x_{k-1} \\ x_{k-2} \\ x_{k-3} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 \\ 1 & -2 & 4 & -8 \\ 1 & -3 & 9 & -27 \end{pmatrix} \cdot \begin{pmatrix} x_k \\ h_{\text{nuevo}} \cdot \dot{x}_k \\ \frac{h_{\text{nuevo}}^2}{2} \cdot \ddot{x}_k \\ \frac{h_{\text{nuevo}}^3}{6} \cdot x_k^{(iii)} \end{pmatrix}$$

Obtenemos otro vector con muestreo equidistante expresado en el nuevo paso h_{nuevo}.

El Control del Paso III

Ahora es fácil cambiar el paso:

$$\begin{pmatrix} x_k \\ h_{\text{nuevo}} \cdot \dot{x}_k \\ \frac{h_{\text{nuevo}}^2}{2} \cdot \ddot{x}_k \\ \frac{h_{\text{nuevo}}^3}{6} \cdot x_k^{(iii)} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{h_{\text{nuevo}}}{h_{\text{viejo}}} & 0 & 0 \\ 0 & 0 & \left(\frac{h_{\text{nuevo}}}{h_{\text{viejo}}}\right)^2 & 0 \\ 0 & 0 & 0 & \left(\frac{h_{\text{nuevo}}}{h_{\text{viejo}}}\right)^3 \end{pmatrix} \cdot \begin{pmatrix} x_k \\ h_{\text{viejo}} \cdot \dot{x}_k \\ \frac{h_{\text{viejo}}^2}{2} \cdot \ddot{x}_k \\ \frac{h_{\text{viejo}}^3}{6} \cdot x_k^{(iii)} \end{pmatrix}$$

Ahora tenemos el vector de Nordsieck expresado en el nuevo paso h_{nuevo}.

En consecuencia se puede escribir:

$$\begin{pmatrix} x_k \\ x_{k-1} \\ x_{k-2} \\ x_{k-3} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 \\ 1 & -2 & 4 & -8 \\ 1 & -3 & 9 & -27 \end{pmatrix} \cdot \begin{pmatrix} x_k \\ h_{\text{nuevo}} \cdot \dot{x}_k \\ \frac{h_{\text{nuevo}}^2}{2} \cdot \ddot{x}_k \\ \frac{h_{\text{nuevo}}^3}{6} \cdot x_k^{(iii)} \end{pmatrix}$$

Obtenemos otro vector con muestreo equidistante expresado en el nuevo paso h_{nuevo}.

No se perdió precisión en el proceso. El nuevo "vector histórico" es del mismo orden de aproximación que el viejo.

El Control del Paso IV

El Control del Paso IV

- Para cambiar el paso de integración, el *vector histórico* de cada variable del estado tiene que multiplicarse consecutivamente por tres matrices.

El Control del Paso IV

- Para cambiar el paso de integración, el *vector histórico* de cada variable del estado tiene que multiplicarse consecutivamente por tres matrices.
- Por eso, cambiar el tamaño del paso durante la simulación usando algoritmos en múltiples pasos es *bastante caro*.

El Control del Paso IV

- Para cambiar el paso de integración, el *vector histórico* de cada variable del estado tiene que multiplicarse consecutivamente por tres matrices.
- Por eso, cambiar el tamaño del paso durante la simulación usando algoritmos en múltiples pasos es *bastante caro*.
- Un *algoritmo para el control del paso* como el de Gustafsson no puede implementarse para algoritmos en múltiples pasos de forma económica.

El Control del Paso IV

- Para cambiar el paso de integración, el *vector histórico* de cada variable del estado tiene que multiplicarse consecutivamente por tres matrices.
- Por eso, cambiar el tamaño del paso durante la simulación usando algoritmos en múltiples pasos es *bastante caro*.
- Un *algoritmo para el control del paso* como el de Gustafsson no puede implementarse para algoritmos en múltiples pasos de forma económica.
- Se prefiere simular usando un *paso más conservador* para evitar que el tamaño del paso tenga que cambiarse frecuentemente.

El Control del Paso IV

- Para cambiar el paso de integración, el *vector histórico* de cada variable del estado tiene que multiplicarse consecutivamente por tres matrices.
- Por eso, cambiar el tamaño del paso durante la simulación usando algoritmos en múltiples pasos es *bastante caro*.
- Un *algoritmo para el control del paso* como el de Gustafsson no puede implementarse para algoritmos en múltiples pasos de forma económica.
- Se prefiere simular usando un *paso más conservador* para evitar que el tamaño del paso tenga que cambiarse frecuentemente.
- Por todas estas razones los *algoritmos de Adams no son competidores* para la simulación de sistemas de la ingeniería.

El Control del Paso IV

- Para cambiar el paso de integración, el *vector histórico* de cada variable del estado tiene que multiplicarse consecutivamente por tres matrices.
- Por eso, cambiar el tamaño del paso durante la simulación usando algoritmos en múltiples pasos es *bastante caro*.
- Un *algoritmo para el control del paso* como el de Gustafsson no puede implementarse para algoritmos en múltiples pasos de forma económica.
- Se prefiere simular usando un *paso más conservador* para evitar que el tamaño del paso tenga que cambiarse frecuentemente.
- Por todas estas razones los *algoritmos de Adams no son competidores* para la simulación de sistemas de la ingeniería.
- Sin embargo, los *algoritmos del tipo BDF son competidores* para la simulación de sistemas de la ingeniería a causa de su capacidad de *tratar con sistemas rígidos*.

El Problema del Arranque

Como ya se explicó, los métodos en múltiples pasos no pueden usarse durante los primeros pasos de la simulación a causa de la necesidad de tener disponible un *vector histórico* para cada variable del estado.

El Problema del Arranque

Como ya se explicó, los métodos en múltiples pasos no pueden usarse durante los primeros pasos de la simulación a causa de la necesidad de tener disponible un *vector histórico* para cada variable del estado.

Una manera muy simple de resolver ese problema con un método que se usa frecuentemente en la práctica, es *cambiar el orden* del algoritmo durante la simulación.

El Problema del Arranque

Como ya se explicó, los métodos en múltiples pasos no pueden usarse durante los primeros pasos de la simulación a causa de la necesidad de tener disponible un *vector histórico* para cada variable del estado.

Una manera muy simple de resolver ese problema con un método que se usa frecuentemente en la práctica, es *cambiar el orden* del algoritmo durante la simulación.

- Empezamos con el algoritmo del orden 1, por ejemplo empezamos con un paso de BDF1. Después del primer paso, ya tenemos un punto histórico.

El Problema del Arranque

Como ya se explicó, los métodos en múltiples pasos no pueden usarse durante los primeros pasos de la simulación a causa de la necesidad de tener disponible un *vector histórico* para cada variable del estado.

Una manera muy simple de resolver ese problema con un método que se usa frecuentemente en la práctica, es *cambiar el orden* del algoritmo durante la simulación.

- Empezamos con el algoritmo del orden 1, por ejemplo empezamos con un paso de BDF1. Después del primer paso, ya tenemos un punto histórico.
- Durante el segundo paso usamos BDF2. Ese paso nos produce un segundo punto histórico.

El Problema del Arranque

Como ya se explicó, los métodos en múltiples pasos no pueden usarse durante los primeros pasos de la simulación a causa de la necesidad de tener disponible un *vector histórico* para cada variable del estado.

Una manera muy simple de resolver ese problema con un método que se usa frecuentemente en la práctica, es *cambiar el orden* del algoritmo durante la simulación.

- Empezamos con el algoritmo del orden 1, por ejemplo empezamos con un paso de BDF1. Después del primer paso, ya tenemos un punto histórico.
- Durante el segundo paso usamos BDF2. Ese paso nos produce un segundo punto histórico.
- Durante el tercer paso usamos BDF3, etc., hasta que llegamos al orden del algoritmo que nos gusta.

El Problema del Arranque

Como ya se explicó, los métodos en múltiples pasos no pueden usarse durante los primeros pasos de la simulación a causa de la necesidad de tener disponible un *vector histórico* para cada variable del estado.

Una manera muy simple de resolver ese problema con un método que se usa frecuentemente en la práctica, es *cambiar el orden* del algoritmo durante la simulación.

- Empezamos con el algoritmo del orden 1, por ejemplo empezamos con un paso de BDF1. Después del primer paso, ya tenemos un punto histórico.
- Durante el segundo paso usamos BDF2. Ese paso nos produce un segundo punto histórico.
- Durante el tercer paso usamos BDF3, etc., hasta que llegamos al orden del algoritmo que nos gusta.

Este método tiene una desventaja importante. Para obtener una precisión aceptable durante los pasos iniciales, tenemos que empezar con pasos muy pequeños. Entonces, una vez llegado al orden deseado tendremos que cambiar el paso.

El Problema del Arranque II

Otro método propuesto en la literatura es usar un algoritmo en un solo paso, por ejemplo un algoritmo RK, del orden deseado durante la iniciación del algoritmo en múltiples pasos.

El Problema del Arranque II

Otro método propuesto en la literatura es usar un algoritmo en un solo paso, por ejemplo un algoritmo RK, del orden deseado durante la iniciación del algoritmo en múltiples pasos.

Por ejemplo, si se quiere simular usando el algoritmo BDF4, empezamos con tres pasos de RK4. Después ya disponemos de los valores históricos que necesitamos para la continuación de la simulación usando el algoritmo BDF4.

El Problema del Arranque II

Otro método propuesto en la literatura es usar un algoritmo en un solo paso, por ejemplo un algoritmo RK, del orden deseado durante la iniciación del algoritmo en múltiples pasos.

Por ejemplo, si se quiere simular usando el algoritmo BDF4, empezamos con tres pasos de RK4. Después ya disponemos de los valores históricos que necesitamos para la continuación de la simulación usando el algoritmo BDF4.

Desafortunadamente, este método tiene la misma desventaja que el método propuesto antes en caso que el sistema que queremos simular usando BDF4 sea rígido. Si no es el caso, no tiene sentido usar BDF4 para su simulación.

El Problema del Arranque II

Otro método propuesto en la literatura es usar un algoritmo en un solo paso, por ejemplo un algoritmo RK, del orden deseado durante la iniciación del algoritmo en múltiples pasos.

Por ejemplo, si se quiere simular usando el algoritmo BDF4, empezamos con tres pasos de RK4. Después ya disponemos de los valores históricos que necesitamos para la continuación de la simulación usando el algoritmo BDF4.

Desafortunadamente, este método tiene la misma desventaja que el método propuesto antes en caso que el sistema que queremos simular usando BDF4 sea rígido. Si no es el caso, no tiene sentido usar BDF4 para su simulación.

Como RK4 es un algoritmo explícito que no sirve para la simulación de sistemas rígidos, tenemos que usar pasos muy pequeños durante la fase inicial de la simulación. Después de la fase inicial tendremos que cambiar el paso.

El Problema de la Salida Densa

Queremos obtener salidas en cualquier instante del tiempo. Como la simulación es cara y como el cambio del tamaño del paso es caro, no nos gusta cambiar la simulación para obtener aproximaciones numéricas de las variables de salida en cualquier instante del tiempo.

El Problema de la Salida Densa

Queremos obtener salidas en cualquier instante del tiempo. Como la simulación es cara y como el cambio del tamaño del paso es caro, no nos gusta cambiar la simulación para obtener aproximaciones numéricas de las variables de salida en cualquier instante del tiempo.

Para ello tendremos que *interpolar* con la misma precisión que usamos durante la simulación. Este concepto se llama la *obtención de salidas densas*.

El Problema de la Salida Densa

Queremos obtener salidas en cualquier instante del tiempo. Como la simulación es cara y como el cambio del tamaño del paso es caro, no nos gusta cambiar la simulación para obtener aproximaciones numéricas de las variables de salida en cualquier instante del tiempo.

Para ello tendremos que *interpolar* con la misma precisión que usamos durante la simulación. Este concepto se llama la *obtención de salidas densas*.

De nuevo nos ayuda el *vector de Nordsieck*. Una vez que pasamos el próximo *instante de comunicación*, es decir, el próximo instante en el cual queremos saber los valores de las salidas, hacemos una "reducción del paso" para obtener una estimación del valor $\mathbf{x_{k-1}}$ en el instante en el cual queremos saber los valores de las salidas.

El Problema de la Salida Densa

Queremos obtener salidas en cualquier instante del tiempo. Como la simulación es cara y como el cambio del tamaño del paso es caro, no nos gusta cambiar la simulación para obtener aproximaciones numéricas de las variables de salida en cualquier instante del tiempo.

Para ello tendremos que *interpolar* con la misma precisión que usamos durante la simulación. Este concepto se llama la *obtención de salidas densas*.

De nuevo nos ayuda el *vector de Nordsieck*. Una vez que pasamos el próximo *instante de comunicación*, es decir, el próximo instante en el cual queremos saber los valores de las salidas, hacemos una "reducción del paso" para obtener una estimación del valor $\mathbf{x_{k-1}}$ en el instante en el cual queremos saber los valores de las salidas.

Es decir, en lugar de reducir el paso para llegar al instante de la comunicación, simplemente continuamos con la simulación usando el paso actual y interpolamos después para obtener una aproximación de las variables de la salida de la misma precisión que hubiéramos obtenido reduciendo el paso.

El Problema de la Salida Densa

Queremos obtener salidas en cualquier instante del tiempo. Como la simulación es cara y como el cambio del tamaño del paso es caro, no nos gusta cambiar la simulación para obtener aproximaciones numéricas de las variables de salida en cualquier instante del tiempo.

Para ello tendremos que *interpolar* con la misma precisión que usamos durante la simulación. Este concepto se llama la *obtención de salidas densas*.

De nuevo nos ayuda el *vector de Nordsieck*. Una vez que pasamos el próximo *instante de comunicación*, es decir, el próximo instante en el cual queremos saber los valores de las salidas, hacemos una "reducción del paso" para obtener una estimación del valor $\mathbf{x_{k-1}}$ en el instante en el cual queremos saber los valores de las salidas.

Es decir, en lugar de reducir el paso para llegar al instante de la comunicación, simplemente continuamos con la simulación usando el paso actual y interpolamos después para obtener una aproximación de las variables de la salida de la misma precisión que hubiéramos obtenido reduciendo el paso.

Después de calcular las salidas continuamos en el último punto de tiempo de la simulación con el paso de antes.

El Problema de la Salida Densa II

Los métodos para la *obtención de salidas densas* también se desarrollaron para *algoritmos en un solo paso*.

El Problema de la Salida Densa II

Los métodos para la *obtención de salidas densas* también se desarrollaron para *algoritmos en un solo paso*.

Desafortunadamente no tenemos acceso al vector de Nordsieck en el caso de algoritmos en un solo paso.

El Problema de la Salida Densa II

Los métodos para la *obtención de salidas densas* también se desarrollaron para *algoritmos en un solo paso*.

Desafortunadamente no tenemos acceso al vector de Nordsieck en el caso de algoritmos en un solo paso.

Ni siquiera podemos usar las aproximaciones intermedias (las predicciones) para estimar las salidas, porque las predicciones normalmente no tienen el mismo orden de aproximación que tiene la corrección.

El Problema de la Salida Densa II

Los métodos para la *obtención de salidas densas* también se desarrollaron para *algoritmos en un solo paso*.

Desafortunadamente no tenemos acceso al vector de Nordsieck en el caso de algoritmos en un solo paso.

Ni siquiera podemos usar las aproximaciones intermedias (las predicciones) para estimar las salidas, porque las predicciones normalmente no tienen el mismo orden de aproximación que tiene la corrección.

La manera que se usa es trabajar con *tres algoritmos empotrados*.

El Problema de la Salida Densa II

Los métodos para la *obtención de salidas densas* también se desarrollaron para *algoritmos en un solo paso*.

Desafortunadamente no tenemos acceso al vector de Nordsieck en el caso de algoritmos en un solo paso.

Ni siquiera podemos usar las aproximaciones intermedias (las predicciones) para estimar las salidas, porque las predicciones normalmente no tienen el mismo orden de aproximación que tiene la corrección.

La manera que se usa es trabajar con *tres algoritmos empotrados*.

Usando las tres aproximaciones de las variables del estado en el tiempo t_{k+1} puede obtenerse una aproximación del mismo orden en otro instante del tiempo $t \in [t_k, t_{k+1}]$.

Conclusiones

En esta presentación enseñamos como pueden encontrarse nuevos algoritmos interesantes de la integración numérica de sistemas dinámicos. Se encontraron nuevos algoritmos de órdenes avanzados para la simulación de sistemas rígidos.

Conclusiones

En esta presentación enseñamos como pueden encontrarse nuevos algoritmos interesantes de la integración numérica de sistemas dinámicos. Se encontraron nuevos algoritmos de órdenes avanzados para la simulación de sistemas rígidos.

A continuación hablamos de los problemas de control de paso y orden, de la iniciación de los algoritmos y de la obtención de salidas densas.

Conclusiones

En esta presentación enseñamos como pueden encontrarse nuevos algoritmos interesantes de la integración numérica de sistemas dinámicos. Se encontraron nuevos algoritmos de órdenes avanzados para la simulación de sistemas rígidos.

A continuación hablamos de los problemas de control de paso y orden, de la iniciación de los algoritmos y de la obtención de salidas densas.

En los códigos "maduros" (códigos de producción) que implementan algún algoritmo de integración en múltiples pasos, el algoritmo mismo ocupa no más que 5% de las instrucciones. Las demás instrucciones sirven para fines de control del algoritmo.