Simulación de Sistemas Continuos y a Tramos

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

27 de junio 2007

Introducción

Hasta ahora analizamos las propiedades de la estabilidad numérica y de la precisión de los algoritmos DAE.

Introducción

Hasta ahora analizamos las propiedades de la estabilidad numérica y de la precisión de los algoritmos DAE.

Ahora discutiremos cuestiones prácticas del desarrollo de códigos que implementan algoritmos DAE.

Introducción

Hasta ahora analizamos las propiedades de la estabilidad numérica y de la precisión de los algoritmos DAE.

Ahora discutiremos cuestiones prácticas del desarrollo de códigos que implementan algoritmos DAE.

Muchas de las dudas concernientes a estos algoritmos que probablemente aún tenéis, se aclararán en el proceso.

El Código DASSL

DASSL es *uno de los códigos de simulación más exitosos* en el mercado de hoy.

El Código DASSL

DASSL es *uno de los códigos de simulación más exitosos* en el mercado de hoy.

DASSL implementa las fórmulas BDF de órdenes 1..5, es decir, se trata de un *código con control del paso y del orden*.

El Código DASSL

DASSL es *uno de los códigos de simulación más exitosos* en el mercado de hoy.

DASSL implementa las fórmulas BDF de órdenes 1..5, es decir, se trata de un *código con control del paso y del orden*.

DASSL fue desarrollado para la simulación de modelos implícitos. Entonces, se trata de un *algoritmo DAE*.

El Código DASSL

DASSL es *uno de los códigos de simulación más exitosos* en el mercado de hoy.

DASSL implementa las fórmulas BDF de órdenes 1..5, es decir, se trata de un *código con control del paso y del orden*.

DASSL fue desarrollado para la simulación de modelos implícitos. Entonces, se trata de un *algoritmo DAE*.

DASSL es el *simulador usado por omisión* en el entorno de modelado y simulación Dymola, es decir, si el usuario no especifica ningún integrador explícitamente, Dymola usará DASSL para la simulación del modelo.

El Código DASSL II

Dymola tiene una preferencia por DASSL a pesar de su *ineficiencia en el tratamiento de modelos no rígidos*.

El Código DASSL II

Dymola tiene una preferencia por DASSL a pesar de su *ineficiencia en el tratamiento de modelos no rígidos*.

Dymola fue desarrollado para la *simulación de modelos grandes*. Casi todos los modelos grandes son rígidos, porque enfocan sobre efectos rápidos y efectos lentos.

El Código DASSL II

Dymola tiene una preferencia por DASSL a pesar de su *ineficiencia en el tratamiento de modelos no rígidos*.

Dymola fue desarrollado para la *simulación de modelos grandes*. Casi todos los modelos grandes son rígidos, porque enfocan sobre efectos rápidos y efectos lentos.

Dymola fue desarrollado para usuarios que no son especialistas. La mayoría de ellos no sabe mucho de los algoritmos de integración numérica. Tampoco saben si sus modelos son rígidos o no lo son. Como los métodos rígidos pueden tratar con modelos no rígidos (aunque no de la forma más eficiente), mientras que los métodos no rígidos no pueden simular modelos rígidos, el uso de DASSL promueve la *robustez del entorno*.

El Código DASSL II

Dymola tiene una preferencia por DASSL a pesar de su *ineficiencia en el tratamiento de modelos no rígidos*.

Dymola fue desarrollado para la *simulación de modelos grandes*. Casi todos los modelos grandes son rígidos, porque enfocan sobre efectos rápidos y efectos lentos.

Dymola fue desarrollado para usuarios que no son especialistas. La mayoría de ellos no sabe mucho de los algoritmos de integración numérica. Tampoco saben si sus modelos son rígidos o no lo son. Como los métodos rígidos pueden tratar con modelos no rígidos (aunque no de la forma más eficiente), mientras que los métodos no rígidos no pueden simular modelos rígidos, el uso de DASSL promueve la *robustez del entorno*.

Las computadoras de hoy son tan poderosas que las consideraciones de optimización de la eficiencia de la simulación han perdido importancia.

El Código DASSL III

Dymola tiene preferencia por DASSL a pesar de su *ineficiencia en el tratamiento de modelos no lineales*.

El Código DASSL III

Dymola tiene preferencia por DASSL a pesar de su *ineficiencia en el tratamiento de modelos no lineales*.

Los modelos no lineales y especialmente *modelos con discontinuidades* exigen *frecuentes cambios del paso* de integración. Como el cambio del paso es caro para los métodos de integración en múltiples pasos, DASSL no es óptimo para la simulación de sistemas con discontinuidades.

El Código DASSL III

Dymola tiene preferencia por DASSL a pesar de su *ineficiencia en el tratamiento de modelos no lineales*.

Los modelos no lineales y especialmente *modelos con discontinuidades* exigen *frecuentes cambios del paso* de integración. Como el cambio del paso es caro para los métodos de integración en múltiples pasos, DASSL no es óptimo para la simulación de sistemas con discontinuidades.

Casi todos los modelos de la ingeniería son modelos con discontinuidades.

El Código DASSL III

Dymola tiene preferencia por DASSL a pesar de su *ineficiencia en el tratamiento de modelos no lineales*.

Los modelos no lineales y especialmente *modelos con discontinuidades* exigen *frecuentes cambios del paso* de integración. Como el cambio del paso es caro para los métodos de integración en múltiples pasos, DASSL no es óptimo para la simulación de sistemas con discontinuidades.

Casi todos los modelos de la ingeniería son modelos con discontinuidades.

Sería entonces preferible usar un código basado en algoritmos del tipo *Radau IIA*.

El Código DASSL III

Dymola tiene preferencia por DASSL a pesar de su *ineficiencia en el tratamiento de modelos no lineales*.

Los modelos no lineales y especialmente *modelos con discontinuidades* exigen *frecuentes cambios del paso* de integración. Como el cambio del paso es caro para los métodos de integración en múltiples pasos, DASSL no es óptimo para la simulación de sistemas con discontinuidades.

Casi todos los modelos de la ingeniería son modelos con discontinuidades.

Sería entonces preferible usar un código basado en algoritmos del tipo *Radau IIA*.

El problema es que aún no existen en el mercado códigos de uso general basados en los métodos IRK. El uso de DASSL entonces promueve la *robustez del entorno*.

El Código DASSL III

Dymola tiene preferencia por DASSL a pesar de su *ineficiencia en el tratamiento de modelos no lineales*.

Los modelos no lineales y especialmente *modelos con discontinuidades* exigen *frecuentes cambios del paso* de integración. Como el cambio del paso es caro para los métodos de integración en múltiples pasos, DASSL no es óptimo para la simulación de sistemas con discontinuidades.

Casi todos los modelos de la ingeniería son modelos con discontinuidades.

Sería entonces preferible usar un código basado en algoritmos del tipo *Radau IIA*.

El problema es que aún no existen en el mercado códigos de uso general basados en los métodos IRK. El uso de DASSL entonces promueve la *robustez del entorno*.

No existen "algoritmos exitosos". Solamente hay códigos exitosos.

El Código DASSL IV

Comparamos otra vez, con un poco más de detalle, la *diferencia entre las formulaciones ODE y DAE* de los algoritmos BDF.

El Código DASSL IV

Comparamos otra vez, con un poco más de detalle, la *diferencia entre las formulaciones ODE y DAE* de los algoritmos BDF.

Empezamos con el sistema lineal explícito:

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

Usamos el algoritmo BDF3 en su forma ODE:

$$\mathbf{x}_{\mathbf{k+1}}^{\mathbf{BDF3}} = \frac{6}{11} h \cdot \dot{\mathbf{x}}_{\mathbf{k+1}} + \frac{18}{11}\mathbf{x}_{\mathbf{k}} - \frac{9}{11}\mathbf{x}_{\mathbf{k-1}} + \frac{2}{11}\mathbf{x}_{\mathbf{k-2}}$$

El Código DASSL IV

Comparamos otra vez, con un poco más de detalle, la *diferencia entre las formulaciones ODE y DAE* de los algoritmos BDF.

Empezamos con el sistema lineal explícito:

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

Usamos el algoritmo BDF3 en su forma ODE:

$$\mathbf{x}_{\mathbf{k+1}}^{\mathbf{BDF3}} = \frac{6}{11} h \cdot \dot{\mathbf{x}}_{\mathbf{k+1}} + \frac{18}{11}\mathbf{x}_{\mathbf{k}} - \frac{9}{11}\mathbf{x}_{\mathbf{k-1}} + \frac{2}{11}\mathbf{x}_{\mathbf{k-2}}$$

Eliminamos las derivadas insertando las ecuaciones del modelo dentro de las ecuaciones del integrador:

$$\mathbf{x}_{\mathbf{k+1}}^{\mathbf{BDF3}} = \frac{6}{11} \cdot \mathbf{A} \cdot h \cdot \mathbf{x}_{\mathbf{k+1}} + \frac{6}{11} \cdot \mathbf{B} \cdot h \cdot \mathbf{u}_{\mathbf{k+1}} + \frac{18}{11}\mathbf{x}_{\mathbf{k}} - \frac{9}{11}\mathbf{x}_{\mathbf{k-1}} + \frac{2}{11}\mathbf{x}_{\mathbf{k-2}}$$

El Código DASSL IV

Comparamos otra vez, con un poco más de detalle, la *diferencia entre las formulaciones ODE y DAE* de los algoritmos BDF.

Empezamos con el sistema lineal explícito:

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

Usamos el algoritmo BDF3 en su forma ODE:

$$\mathbf{x}_{\mathbf{k+1}}^{\mathbf{BDF3}} = \frac{6}{11} h \cdot \dot{\mathbf{x}}_{\mathbf{k+1}} + \frac{18}{11}\mathbf{x}_{\mathbf{k}} - \frac{9}{11}\mathbf{x}_{\mathbf{k-1}} + \frac{2}{11}\mathbf{x}_{\mathbf{k-2}}$$

Eliminamos las derivadas insertando las ecuaciones del modelo dentro de las ecuaciones del integrador:

$$\mathbf{x}_{\mathbf{k+1}}^{\mathbf{BDF3}} = \frac{6}{11} \cdot \mathbf{A} \cdot h \cdot \mathbf{x}_{\mathbf{k+1}} + \frac{6}{11} \cdot \mathbf{B} \cdot h \cdot \mathbf{u}_{\mathbf{k+1}} + \frac{18}{11}\mathbf{x}_{\mathbf{k}} - \frac{9}{11}\mathbf{x}_{\mathbf{k-1}} + \frac{2}{11}\mathbf{x}_{\mathbf{k-2}}$$

Formulamos la iteración de Newton de la forma siguiente:

$$\mathcal{F}(\mathbf{x}_{\mathbf{k+1}})^{\mathbf{ODE}} = \mathbf{x}_{\mathbf{k+1}} - \mathbf{x}_{\mathbf{k+1}}^{\mathbf{BDF3}} = 0.0$$

donde $\mathbf{x}_{\mathbf{k+1}}^{\mathbf{BDF3}}$ es la solución conocida aproximada por el integrador, mientras que $\mathbf{x}_{\mathbf{k+1}}$ es la solución analítica desconocida.

El Código DASSL V

Con:

$$\mathcal{F}(\mathbf{x_{k+1}})^{\mathbf{ODE}} = \mathbf{x_{k+1}} - \mathbf{x}_{\mathbf{k+1}}^{\mathbf{BDF3}}$$

podemos calcular el Hessiano de la iteración:

$$\mathcal{H}(\mathbf{x_{k+1}})^{\mathbf{ODE}} = \frac{\partial \mathcal{F}(\mathbf{x_{k+1}})^{\mathbf{ODE}}}{\partial \mathbf{x_{k+1}}} = \mathbf{I}^{(\mathbf{n})} - \frac{6}{11} \cdot \mathbf{A} \cdot h$$

El Código DASSL V

Con:

$$\mathcal{F}(\mathbf{x_{k+1}})^{\mathbf{ODE}} = \mathbf{x_{k+1}} - \mathbf{x}_{\mathbf{k+1}}^{\mathbf{BDF3}}$$

podemos calcular el Hessiano de la iteración:

$$\mathcal{H}(\mathbf{x_{k+1}})^{\mathbf{ODE}} = \frac{\partial \mathcal{F}(\mathbf{x_{k+1}})^{\mathbf{ODE}}}{\partial \mathbf{x_{k+1}}} = \mathbf{I}^{(\mathbf{n})} - \frac{6}{11} \cdot \mathbf{A} \cdot h$$

En el caso de un sistema no lineal:

$$\mathcal{H}(\mathbf{x_{k+1}})^{\mathbf{ODE}} = \mathbf{I}^{(\mathbf{n})} - \frac{6}{11} \cdot \mathcal{J} \cdot h$$

donde $\mathcal{J}$ es el Jacobiano del sistema:

$$\mathcal{J}(\mathbf{x_{k+1}}) = \frac{\partial \mathbf{f}(\mathbf{x_{k+1}})}{\partial \mathbf{x_{k+1}}}$$

El Código DASSL VI

¿Qué pasa en el caso de la formulación DAE?

El Código DASSL VI

¿Qué pasa en el caso de la formulación DAE?

Usamos el algoritmo BDF resuelto por la derivada:

$$\dot{x}_{k+1}^{BDF3} = \frac{1}{h}\left[\frac{11}{6} \cdot x_{k+1} - 3x_k + \frac{3}{2}x_{k-1} - \frac{1}{3}x_{k-2}\right]$$

El Código DASSL VI

¿Qué pasa en el caso de la formulación DAE?

Usamos el algoritmo BDF resuelto por la derivada:

$$\dot{\mathbf{x}}_{\mathbf{k+1}}^{\mathbf{BDF3}} = \frac{1}{h}\left[\frac{11}{6}\cdot \mathbf{x}_{\mathbf{k+1}} - 3\mathbf{x}_{\mathbf{k}} + \frac{3}{2}\mathbf{x}_{\mathbf{k-1}} - \frac{1}{3}\mathbf{x}_{\mathbf{k-2}}\right]$$

Formulamos ahora la iteración de Newton de la forma siguiente:

$$\mathcal{F}(\mathbf{x}_{\mathbf{k+1}})^{\mathbf{DAE}} = \dot{\mathbf{x}}_{\mathbf{k+1}}^{\mathbf{modelo}} - \dot{\mathbf{x}}_{\mathbf{k+1}}^{\mathbf{BDF3}} = 0.0$$

donde $\dot{\mathbf{x}}_{\mathbf{k+1}}^{\mathbf{BDF3}}$ es la derivada aproximada por el integrador, mientras que $\dot{\mathbf{x}}_{\mathbf{k+1}}^{\mathbf{modelo}}$ es la derivada obtenida del modelo.

El Código DASSL VI

¿Qué pasa en el caso de la formulación DAE?

Usamos el algoritmo BDF resuelto por la derivada:

$$\dot{\mathbf{x}}_{\mathbf{k+1}}^{\mathbf{BDF3}} = \frac{1}{h}\left[\frac{11}{6}\cdot \mathbf{x}_{\mathbf{k+1}} - 3\mathbf{x}_{\mathbf{k}} + \frac{3}{2}\mathbf{x}_{\mathbf{k-1}} - \frac{1}{3}\mathbf{x}_{\mathbf{k-2}}\right]$$

Formulamos ahora la iteración de Newton de la forma siguiente:

$$\mathcal{F}(\mathbf{x}_{\mathbf{k+1}})^{\mathbf{DAE}} = \dot{\mathbf{x}}_{\mathbf{k+1}}^{\mathbf{modelo}} - \dot{\mathbf{x}}_{\mathbf{k+1}}^{\mathbf{BDF3}} = 0.0$$

donde $\dot{\mathbf{x}}_{\mathbf{k+1}}^{\mathbf{BDF3}}$ es la derivada aproximada por el integrador, mientras que $\dot{\mathbf{x}}_{\mathbf{k+1}}^{\mathbf{modelo}}$ es la derivada obtenida del modelo.

Entonces:

$$\mathcal{H}(\mathbf{x}_{\mathbf{k+1}})^{\mathbf{DAE}} = \frac{\partial \mathcal{F}(\mathbf{x}_{\mathbf{k+1}})^{\mathbf{DAE}}}{\partial \mathbf{x}_{\mathbf{k+1}}} = \mathbf{A} - \frac{11}{6h}\cdot \mathbf{I}^{(\mathbf{n})}$$

El Código DASSL VI

¿Qué pasa en el caso de la formulación DAE?

Usamos el algoritmo BDF resuelto por la derivada:

$$\dot{\mathbf{x}}_{\mathbf{k+1}}^{\mathbf{BDF3}} = \frac{1}{h}\left[\frac{11}{6}\cdot \mathbf{x}_{\mathbf{k+1}} - 3\mathbf{x}_{\mathbf{k}} + \frac{3}{2}\mathbf{x}_{\mathbf{k-1}} - \frac{1}{3}\mathbf{x}_{\mathbf{k-2}}\right]$$

Formulamos ahora la iteración de Newton de la forma siguiente:

$$\mathcal{F}(\mathbf{x}_{\mathbf{k+1}})^{\mathbf{DAE}} = \dot{\mathbf{x}}_{\mathbf{k+1}}^{\mathbf{modelo}} - \dot{\mathbf{x}}_{\mathbf{k+1}}^{\mathbf{BDF3}} = 0.0$$

donde $\dot{\mathbf{x}}_{\mathbf{k+1}}^{\mathbf{BDF3}}$ es la derivada aproximada por el integrador, mientras que $\dot{\mathbf{x}}_{\mathbf{k+1}}^{\mathbf{modelo}}$ es la derivada obtenida del modelo.

Entonces:

$$\mathcal{H}(\mathbf{x}_{\mathbf{k+1}})^{\mathbf{DAE}} = \frac{\partial \mathcal{F}(\mathbf{x}_{\mathbf{k+1}})^{\mathbf{DAE}}}{\partial \mathbf{x}_{\mathbf{k+1}}} = \mathbf{A} - \frac{11}{6h}\cdot \mathbf{I}^{(\mathbf{n})}$$

En el caso de un sistema no lineal:

$$\mathcal{H}(\mathbf{x}_{\mathbf{k+1}})^{\mathbf{DAE}} = \mathcal{J} - \frac{11}{6h}\cdot \mathbf{I}^{(\mathbf{n})}$$

El Código DASSL VII

¿Qué pasa si tenemos que usar pasos de integración muy pequeños?

El Código DASSL VII

¿Qué pasa si tenemos que usar pasos de integración muy pequeños?

En el caso de la *formulación ODE*:

$$\lim_{h \longrightarrow 0} \mathcal{H}(\mathbf{x_{k+1}})^{\mathbf{ODE}} = \lim_{h \longrightarrow 0} \left(\mathbf{I^{(n)}} - \frac{6}{11} \cdot \mathcal{J} \cdot h \right) = \mathbf{I^{(n)}}$$

El Código DASSL VII

¿Qué pasa si tenemos que usar pasos de integración muy pequeños?

En el caso de la *formulación ODE*:

$$\lim_{h \longrightarrow 0} \mathcal{H}(\mathbf{x_{k+1}})^{\mathbf{ODE}} = \lim_{h \longrightarrow 0} \left(\mathbf{I^{(n)}} - \frac{6}{11} \cdot \mathcal{J} \cdot h \right) = \mathbf{I^{(n)}}$$

En el caso de la *formulación DAE*:

$$\lim_{h \longrightarrow 0} \mathcal{H}(\mathbf{x_{k+1}})^{\mathbf{DAE}} = \lim_{h \longrightarrow 0} \left(\mathcal{J} - \frac{11}{6h} \cdot \mathbf{I^{(n)}} \right) \rightarrow \infty$$

El Código DASSL VII

¿Qué pasa si tenemos que usar pasos de integración muy pequeños?

En el caso de la *formulación ODE*:

$$\lim_{h \to 0} \mathcal{H}(\mathbf{x_{k+1}})^{\mathbf{ODE}} = \lim_{h \to 0} \left(\mathbf{I^{(n)}} - \frac{6}{11} \cdot \mathcal{J} \cdot h \right) = \mathbf{I^{(n)}}$$

En el caso de la *formulación DAE*:

$$\lim_{h \to 0} \mathcal{H}(\mathbf{x_{k+1}})^{\mathbf{DAE}} = \lim_{h \to 0} \left(\mathcal{J} - \frac{11}{6h} \cdot \mathbf{I^{(n)}} \right) \to \infty$$

En la formulación DAE con pasos de integración pequeños, el Hessiano de la iteración es muy sensible a los cambios del paso. Eso nos hace pensar que posiblemente tendremos problemas con el algoritmo.

El Código DASSL VIII

¿Qué pasa en el caso de un modelo implícito?

El Código DASSL VIII

¿Qué pasa en el caso de un modelo implícito?

Veamos otra vez las fórmulas BDF:

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

$$\mathbf{x_{k+1}} = \frac{2}{3} \cdot h \cdot \mathbf{f_{k+1}} + \frac{4}{3} \cdot \mathbf{x_k} - \frac{1}{3} \cdot \mathbf{x_{k-1}}$$

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

$$\text{etc.}$$

El Código DASSL VIII

¿Qué pasa en el caso de un modelo implícito?

Veamos otra vez las fórmulas BDF:

$$\mathbf{x_{k+1}} = h \cdot \mathbf{f_{k+1}} + \mathbf{x_k}$$
$$\mathbf{x_{k+1}} = \frac{2}{3} \cdot h \cdot \mathbf{f_{k+1}} + \frac{4}{3} \cdot \mathbf{x_k} - \frac{1}{3} \cdot \mathbf{x_{k-1}}$$
$$\mathbf{x_{k+1}} = \frac{6}{11} \cdot h \cdot \mathbf{f_{k+1}} + \frac{18}{11} \cdot \mathbf{x_k} - \frac{9}{11} \cdot \mathbf{x_{k-1}} + \frac{2}{11} \cdot \mathbf{x_{k-2}}$$
$$\text{etc.}$$

Podemos generalizar estas fórmulas:

$$\mathbf{x_{k+1}} = \bar{h} \cdot \mathbf{f_{k+1}} + \text{viejo}(\mathbf{x})$$

donde $\bar{h}$ es un *paso normalizado* proporcional al paso h y $\text{viejo}(\mathbf{x})$ es una función de información del pasado que no influye en la iteración de Newton.

El Código DASSL VIII

¿Qué pasa en el caso de un modelo implícito?

Veamos otra vez las fórmulas BDF:

$$\mathbf{x_{k+1}} = h \cdot \mathbf{f_{k+1}} + \mathbf{x_k}$$
$$\mathbf{x_{k+1}} = \frac{2}{3} \cdot h \cdot \mathbf{f_{k+1}} + \frac{4}{3} \cdot \mathbf{x_k} - \frac{1}{3} \cdot \mathbf{x_{k-1}}$$
$$\mathbf{x_{k+1}} = \frac{6}{11} \cdot h \cdot \mathbf{f_{k+1}} + \frac{18}{11} \cdot \mathbf{x_k} - \frac{9}{11} \cdot \mathbf{x_{k-1}} + \frac{2}{11} \cdot \mathbf{x_{k-2}}$$
$$\text{etc.}$$

Podemos generalizar estas fórmulas:

$$\mathbf{x_{k+1}} = \bar{h} \cdot \mathbf{f_{k+1}} + \text{viejo}(\mathbf{x})$$

donde $\bar{h}$ es un *paso normalizado* proporcional al paso h y $\text{viejo}(\mathbf{x})$ es una función de información del pasado que no influye en la iteración de Newton.

Entonces:

$$\mathbf{f_{k+1}} = \frac{\mathbf{x_{k+1}} - \text{viejo}(\mathbf{x})}{\bar{h}}$$

El Código DASSL IX

Consideramos ahora el modelo implícito:

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

El Código DASSL IX

Consideramos ahora el modelo implícito:

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

Podemos eliminar las derivadas insertando la fórmula del integrador en el modelo implícito:

$$\mathcal{F}(\mathbf{x_{k+1}}) = \mathbf{f}(\mathbf{x_{k+1}}, \frac{\mathbf{x_{k+1}} - \text{viejo}(\mathbf{x})}{\bar{h}}, \mathbf{u_{k+1}}, t_{k+1}) = 0.0$$

El Código DASSL IX

Consideramos ahora el modelo implícito:

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

Podemos eliminar las derivadas insertando la fórmula del integrador en el modelo implícito:

$$\mathcal{F}(\mathbf{x_{k+1}}) = \mathbf{f}(\mathbf{x_{k+1}}, \frac{\mathbf{x_{k+1}} - \text{viejo}(\mathbf{x})}{\bar{h}}, \mathbf{u_{k+1}}, t_{k+1}) = 0.0$$

En realidad no es necesario insertar físicamente el algoritmo de la integración en el modelo, porque podemos ver fácilmente, cuales serán las implicaciones de tal inserción.

El Código DASSL IX

Consideramos ahora el modelo implícito:

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

Podemos eliminar las derivadas insertando la fórmula del integrador en el modelo implícito:

$$\mathcal{F}(\mathbf{x_{k+1}}) = \mathbf{f}(\mathbf{x_{k+1}}, \frac{\mathbf{x_{k+1}} - \text{viejo}(\mathbf{x})}{\bar{h}}, \mathbf{u_{k+1}}, t_{k+1}) = 0.0$$

En realidad no es necesario insertar físicamente el algoritmo de la integración en el modelo, porque podemos ver fácilmente, cuales serán las implicaciones de tal inserción.

Entonces:

$$\mathcal{H}(\mathbf{x_{k+1}}) = \mathcal{J}_{\mathbf{x}}(\mathbf{x_{k+1}}) + \frac{1}{\bar{h}} \cdot \mathcal{J}_{\dot{\mathbf{x}}}(\mathbf{x_{k+1}})$$

donde:

$$\mathcal{J}_{\mathbf{x}}(\mathbf{x_{k+1}}) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{x=x_{k+1}, \dot{x}=\dot{x}_{k+1}}$$

$$\mathcal{J}_{\dot{\mathbf{x}}}(\mathbf{x_{k+1}}) = \left.\frac{\partial \mathbf{f}}{\partial \dot{\mathbf{x}}}\right|_{x=x_{k+1}, \dot{x}=\dot{x}_{k+1}}$$

El Código DASSL X

Podemos implementar la iteración de Newton de la forma siguiente:

$$\left(\mathcal{J}_{\mathbf{x}} + \frac{1}{\bar{h}} \cdot \mathcal{J}_{\dot{\mathbf{x}}}\right) \cdot \delta^{\ell} = \mathbf{f}(\mathbf{x}^{\ell}, \dot{\mathbf{x}}^{\ell}, \mathbf{u}, t)$$

$$\mathbf{x}^{\ell+1} = \mathbf{x}^{\ell} - \delta^{\ell}$$

$$\dot{\mathbf{x}}^{\ell+1} = \dot{\mathbf{x}}^{\ell} - \frac{1}{\bar{h}} \cdot \delta^{\ell}$$

El Código DASSL X

Podemos implementar la iteración de Newton de la forma siguiente:

$$\left(\mathcal{J}_{\mathbf{x}} + \frac{1}{\bar{h}} \cdot \mathcal{J}_{\dot{\mathbf{x}}}\right) \cdot \delta^{\ell} = \mathbf{f}(\mathbf{x}^{\ell}, \dot{\mathbf{x}}^{\ell}, \mathbf{u}, t)$$
$$\mathbf{x}^{\ell+1} = \mathbf{x}^{\ell} - \delta^{\ell}$$
$$\dot{\mathbf{x}}^{\ell+1} = \dot{\mathbf{x}}^{\ell} - \frac{1}{\bar{h}} \cdot \delta^{\ell}$$

o aún mejor:

$$\left(\bar{h} \cdot \mathcal{J}_{\mathbf{x}} + \mathcal{J}_{\dot{\mathbf{x}}}\right) \cdot \delta^{\ell} = \bar{h} \cdot \mathbf{f}(\mathbf{x}^{\ell}, \dot{\mathbf{x}}^{\ell}, \mathbf{u}, t)$$
$$\mathbf{x}^{\ell+1} = \mathbf{x}^{\ell} - \delta^{\ell}$$
$$\dot{\mathbf{x}}^{\ell+1} = \dot{\mathbf{x}}^{\ell} - \frac{1}{\bar{h}} \cdot \delta^{\ell}$$

El Código DASSL X

Podemos implementar la iteración de Newton de la forma siguiente:

$$\left(\mathcal{J}_{\mathbf{x}} + \frac{1}{\bar{h}} \cdot \mathcal{J}_{\dot{\mathbf{x}}}\right) \cdot \delta^{\ell} = \mathbf{f}(\mathbf{x}^{\ell}, \dot{\mathbf{x}}^{\ell}, \mathbf{u}, t)$$
$$\mathbf{x}^{\ell+1} = \mathbf{x}^{\ell} - \delta^{\ell}$$
$$\dot{\mathbf{x}}^{\ell+1} = \dot{\mathbf{x}}^{\ell} - \frac{1}{\bar{h}} \cdot \delta^{\ell}$$

o aún mejor:

$$(\bar{h} \cdot \mathcal{J}_{\mathbf{x}} + \mathcal{J}_{\dot{\mathbf{x}}}) \cdot \delta^{\ell} = \bar{h} \cdot \mathbf{f}(\mathbf{x}^{\ell}, \dot{\mathbf{x}}^{\ell}, \mathbf{u}, t)$$
$$\mathbf{x}^{\ell+1} = \mathbf{x}^{\ell} - \delta^{\ell}$$
$$\dot{\mathbf{x}}^{\ell+1} = \dot{\mathbf{x}}^{\ell} - \frac{1}{\bar{h}} \cdot \delta^{\ell}$$

En cada paso de la iteración tenemos que resolver un sistema lineal.

El Código DASSL XI

¿Qué pasa en el caso de pasos de integración pequeños?

El Código DASSL XI

¿Qué pasa en el caso de pasos de integración pequeños?

Puede verse que:

$$\lim_{\bar{h} \longrightarrow 0} \left(\bar{h} \cdot \mathcal{J}_{\mathbf{x}} + \mathcal{J}_{\dot{\mathbf{x}}} \right) = \mathcal{J}_{\dot{\mathbf{x}}}$$

El Código DASSL XI

¿Qué pasa en el caso de pasos de integración pequeños?

Puede verse que:

$$\lim_{\bar{h}\longrightarrow 0}\left(\bar{h}\cdot\mathcal{J}_{\mathbf{x}}+\mathcal{J}_{\dot{\mathbf{x}}}\right)=\mathcal{J}_{\dot{\mathbf{x}}}$$

Sin embargo, ya sabemos de la presentación anterior que la matriz $\mathcal{J}_{\dot{\mathbf{x}}}$ es *singular* en el caso de *modelos de alto índice*, porque en el caso lineal:

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

sabemos que $\mathcal{J}_{\mathbf{x}}=\mathbf{A}$ y $\mathcal{J}_{\dot{\mathbf{x}}}=\mathbf{B}$.

El Código DASSL XI

¿Qué pasa en el caso de pasos de integración pequeños?

Puede verse que:

$$\lim_{\bar{h}\longrightarrow 0}\left(\bar{h}\cdot\mathcal{J}_{\mathbf{x}}+\mathcal{J}_{\dot{\mathbf{x}}}\right)=\mathcal{J}_{\dot{\mathbf{x}}}$$

Sin embargo, ya sabemos de la presentación anterior que la matriz $\mathcal{J}_{\dot{\mathbf{x}}}$ es *singular* en el caso de *modelos de alto índice*, porque en el caso lineal:

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

sabemos que $\mathcal{J}_{\mathbf{x}}=\mathbf{A}$ y $\mathcal{J}_{\dot{\mathbf{x}}}=\mathbf{B}$.

DASSL no sirve para la simulación de modelos DAE de alto índice.

El Código DASSL XII

Suponemos ahora que tenemos un modelo con n ecuaciones diferenciales y k ecuaciones algebraicas. El modelo:

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

contiene $n + k$ ecuaciones en $n + k$ variables.

El Código DASSL XII

Suponemos ahora que tenemos un modelo con n ecuaciones diferenciales y k ecuaciones algebraicas. El modelo:

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

contiene $n + k$ ecuaciones en $n + k$ variables.

No se trata de un sistema de alto índice, pero la matriz $\mathcal{J}_{\dot{\mathbf{x}}}$ aun es singular.

El Código DASSL XII

Suponemos ahora que tenemos un modelo con n ecuaciones diferenciales y k ecuaciones algebraicas. El modelo:

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

contiene $n + k$ ecuaciones en $n + k$ variables.

No se trata de un sistema de alto índice, pero la matriz $\mathcal{J}_{\dot{\mathbf{x}}}$ aun es singular.

DASSL también tiene problemas con la simulación de sistemas del índice 1 para pasos de la integración muy pequeños.

El Código DASSL XIII

Por estas razones:

El Código DASSL XIII

Por estas razones:

- Dymola no usa DASSL como *algoritmo DAE*, sino como *algoritmo ODE*.

El Código DASSL XIII

Por estas razones:

- Dymola no usa DASSL como *algoritmo DAE*, sino como *algoritmo ODE*.
- Durante la compilación del modelo, Dymola primero reduce el índice del modelo a 1 usando el *algoritmo de Pantelides*.

El Código DASSL XIII

Por estas razones:

- Dymola no usa DASSL como *algoritmo DAE*, sino como *algoritmo ODE*.
- Durante la compilación del modelo, Dymola primero reduce el índice del modelo a 1 usando el *algoritmo de Pantelides*.
- Después, Dymola identifica las variables de iteración que necesita para resolver todos los *bucles algebraicos*.

El Código DASSL XIII

Por estas razones:

- Dymola no usa DASSL como *algoritmo DAE*, sino como *algoritmo ODE*.
- Durante la compilación del modelo, Dymola primero reduce el índice del modelo a 1 usando el *algoritmo de Pantelides*.
- Después, Dymola identifica las variables de iteración que necesita para resolver todos los *bucles algebraicos*.
- Para ello, Dymola usa el *algoritmo de Tarjan* para aislar los bucles algebraicos y minimizar sus tamaños.

El Código DASSL XIII

Por estas razones:

- Dymola no usa DASSL como *algoritmo DAE*, sino como *algoritmo ODE*.
- Durante la compilación del modelo, Dymola primero reduce el índice del modelo a 1 usando el *algoritmo de Pantelides*.
- Después, Dymola identifica las variables de iteración que necesita para resolver todos los *bucles algebraicos*.
- Para ello, Dymola usa el *algoritmo de Tarjan* para aislar los bucles algebraicos y minimizar sus tamaños.
- Después, Dymola usa el *algoritmo de rasgadura* para encontrar un número pequeño de variables de iteración.

El Código DASSL XIII

Por estas razones:

- Dymola no usa DASSL como *algoritmo DAE*, sino como *algoritmo ODE*.
- Durante la compilación del modelo, Dymola primero reduce el índice del modelo a 1 usando el *algoritmo de Pantelides*.
- Después, Dymola identifica las variables de iteración que necesita para resolver todos los *bucles algebraicos*.
- Para ello, Dymola usa el *algoritmo de Tarjan* para aislar los bucles algebraicos y minimizar sus tamaños.
- Después, Dymola usa el *algoritmo de rasgadura* para encontrar un número pequeño de variables de iteración.
- Durante la simulación, Dymola usa DASSL como *algoritmo ODE*. DASSL itera sobre las n variables del estado.

El Código DASSL XIII

Por estas razones:

- Dymola no usa DASSL como *algoritmo DAE*, sino como *algoritmo ODE*.
- Durante la compilación del modelo, Dymola primero reduce el índice del modelo a 1 usando el *algoritmo de Pantelides*.
- Después, Dymola identifica las variables de iteración que necesita para resolver todos los *bucles algebraicos*.
- Para ello, Dymola usa el *algoritmo de Tarjan* para aislar los bucles algebraicos y minimizar sus tamaños.
- Después, Dymola usa el *algoritmo de rasgadura* para encontrar un número pequeño de variables de iteración.
- Durante la simulación, Dymola usa DASSL como *algoritmo ODE*. DASSL itera sobre las n variables del estado.
- Dentro de cada paso de la iteración, Dymola evalúa el modelo. En la descripción del modelo se encuentran pequeños *bucles algebraicos*. Cada uno entre ellos invoca una iteración de Newton para encontrar los valores actuales de las *variables de rasgadura*.

La Integración en Línea

La Integración en Línea

Hasta ahora distinguimos siempre entre las ecuaciones del modelo y las del integrador. La razón era simple. Los usuarios en general saben poco de los métodos numéricos de la integración y por eso prefieren usar el código de integración como una caja negra.

La Integración en Línea

Hasta ahora distinguimos siempre entre las ecuaciones del modelo y las del integrador. La razón era simple. Los usuarios en general saben poco de los métodos numéricos de la integración y por eso prefieren usar el código de integración como una caja negra.

Es factible que puedan obtenerse simulaciones más económicas si no se hace esta distinción.

La Integración en Línea

Hasta ahora distinguimos siempre entre las ecuaciones del modelo y las del integrador. La razón era simple. Los usuarios en general saben poco de los métodos numéricos de la integración y por eso prefieren usar el código de integración como una caja negra.

Es factible que puedan obtenerse simulaciones más económicas si no se hace esta distinción.

La *integración en línea* mezcla los dos tipos de ecuaciones. Las ecuaciones del algoritmo simplemente se añaden al conjunto de las ecuaciones del modelo.

La Integración en Línea

Hasta ahora distinguimos siempre entre las ecuaciones del modelo y las del integrador. La razón era simple. Los usuarios en general saben poco de los métodos numéricos de la integración y por eso prefieren usar el código de integración como una caja negra.

Es factible que puedan obtenerse simulaciones más económicas si no se hace esta distinción.

La *integración en línea* mezcla los dos tipos de ecuaciones. Las ecuaciones del algoritmo simplemente se añaden al conjunto de las ecuaciones del modelo.

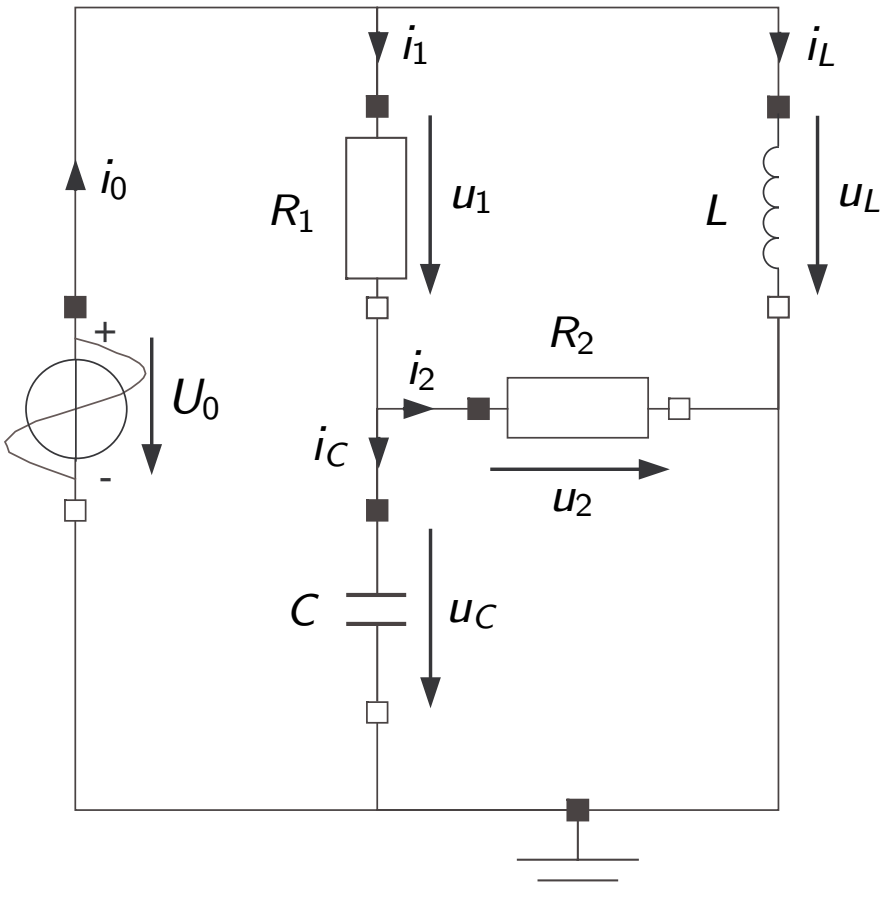
De esta manera, el *sistema de ecuaciones diferenciales algebraicas* se convierte en un *sistema de ecuaciones en diferencias y ecuaciones algebraicas*.

La Integración en Línea II

Empezamos con un ejemplo. Se simula un pequeño circuito eléctrico.

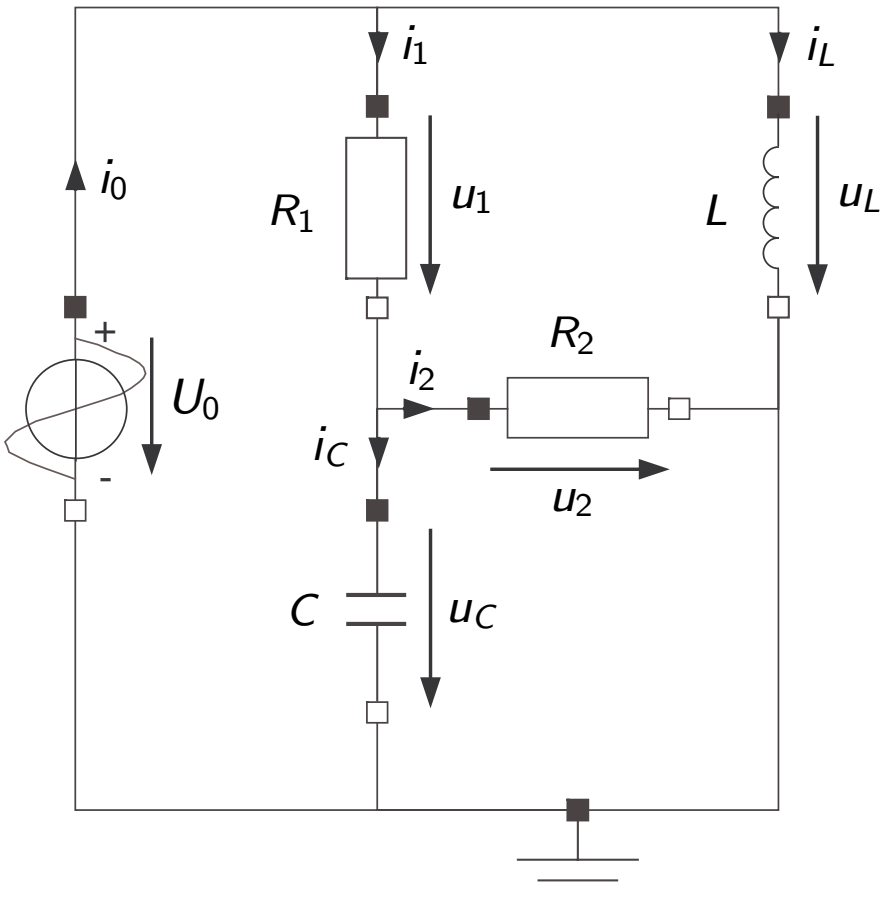
La Integración en Línea II

Empezamos con un ejemplo. Se simula un pequeño circuito eléctrico.



La Integración en Línea II

Empezamos con un ejemplo. Se simula un pequeño circuito eléctrico.



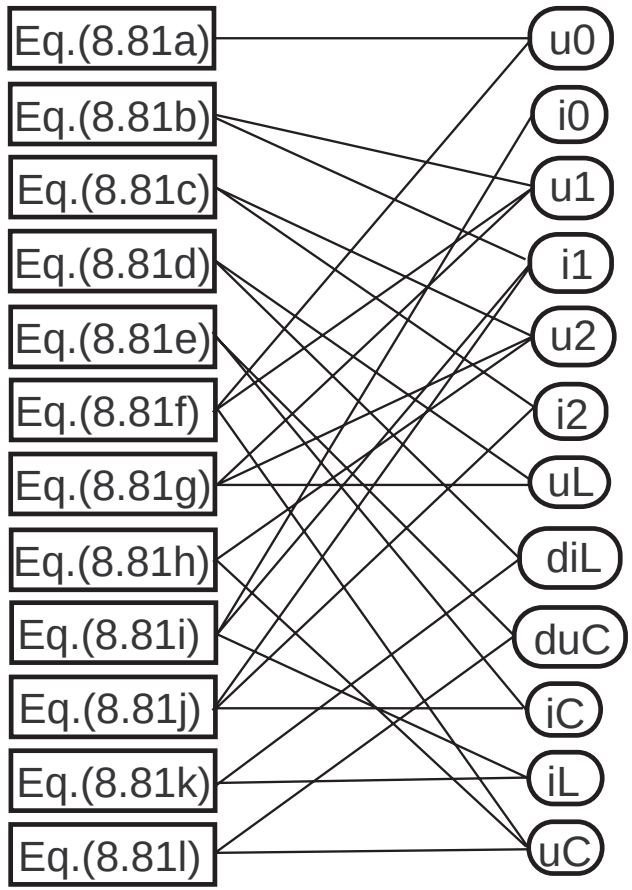
$$u_0 = f(t)$$
$$u_1 = R_1 \cdot i_1$$
$$u_2 = R_2 \cdot i_2$$
$$u_L = L \cdot di_L$$
$$i_C = C \cdot du_C$$
$$u_0 = u_1 + u_C$$
$$u_L = u_1 + u_2$$
$$u_C = u_2$$
$$i_0 = i_1 + i_L$$
$$i_1 = i_2 + i_C$$
$$i_L = \text{viejo}(i_L) + \bar{h} \cdot di_L$$
$$u_C = \text{viejo}(u_C) + \bar{h} \cdot du_C$$

El Dígrafo de la Estructura

$$
\begin{aligned}
u_0 &= f(t) \\
u_1 &= R_1 \cdot i_1 \\
u_2 &= R_2 \cdot i_2 \\
u_L &= L \cdot di_L \\
i_C &= C \cdot du_C \\
u_0 &= u_1 + u_C \\
u_L &= u_1 + u_2 \\
u_C &= u_2 \\
i_0 &= i_1 + i_L \\
i_1 &= i_2 + i_C \\
i_L &= \text{viejo}(i_L) + \bar{h} \cdot di_L \\
u_C &= \text{viejo}(u_C) + \bar{h} \cdot du_C
\end{aligned}
$$

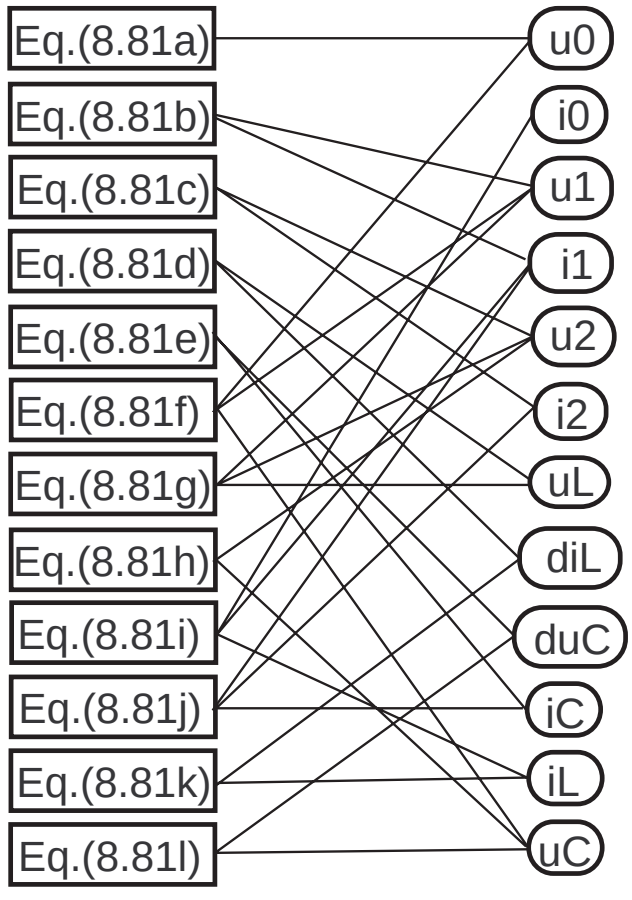
El Dígrafo de la Estructura

$$u_0 = f(t)$$
$$u_1 = R_1 \cdot i_1$$
$$u_2 = R_2 \cdot i_2$$
$$u_L = L \cdot di_L$$
$$i_C = C \cdot du_C$$
$$u_0 = u_1 + u_C$$
$$u_L = u_1 + u_2$$
$$u_C = u_2$$
$$i_0 = i_1 + i_L$$
$$i_1 = i_2 + i_C$$
$$i_L = \text{viejo}(i_L) + \bar{h} \cdot di_L$$
$$u_C = \text{viejo}(u_C) + \bar{h} \cdot du_C$$



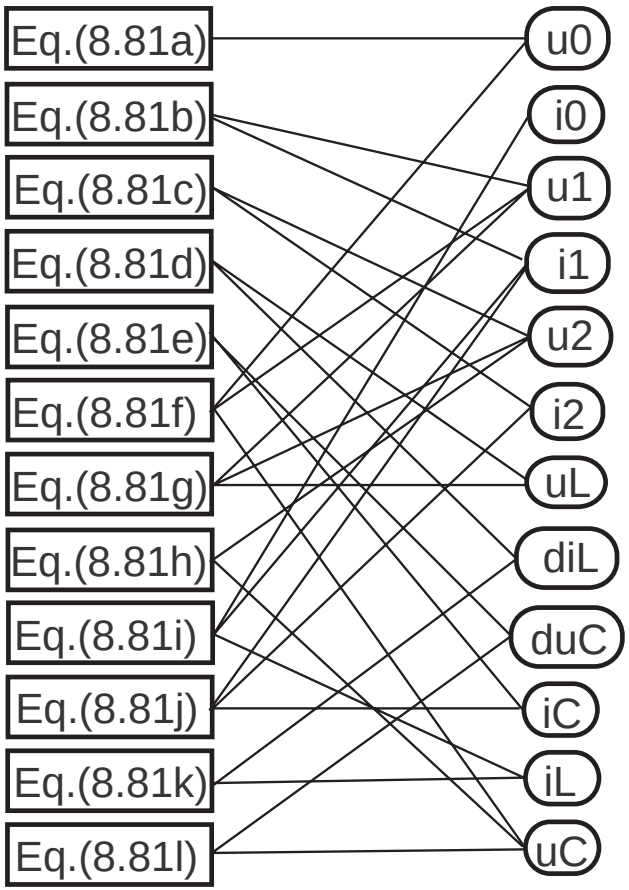
El Dígrafo de la Estructura

$$u_0 = f(t)$$
$$u_1 = R_1 \cdot i_1$$
$$u_2 = R_2 \cdot i_2$$
$$u_L = L \cdot di_L$$
$$i_C = C \cdot du_C$$
$$u_0 = u_1 + u_C$$
$$u_L = u_1 + u_2$$
$$u_C = u_2$$
$$i_0 = i_1 + i_L$$
$$i_1 = i_2 + i_C$$
$$i_L = \text{viejo}(i_L) + \bar{h} \cdot di_L$$
$$u_C = \text{viejo}(u_C) + \bar{h} \cdot du_C$$

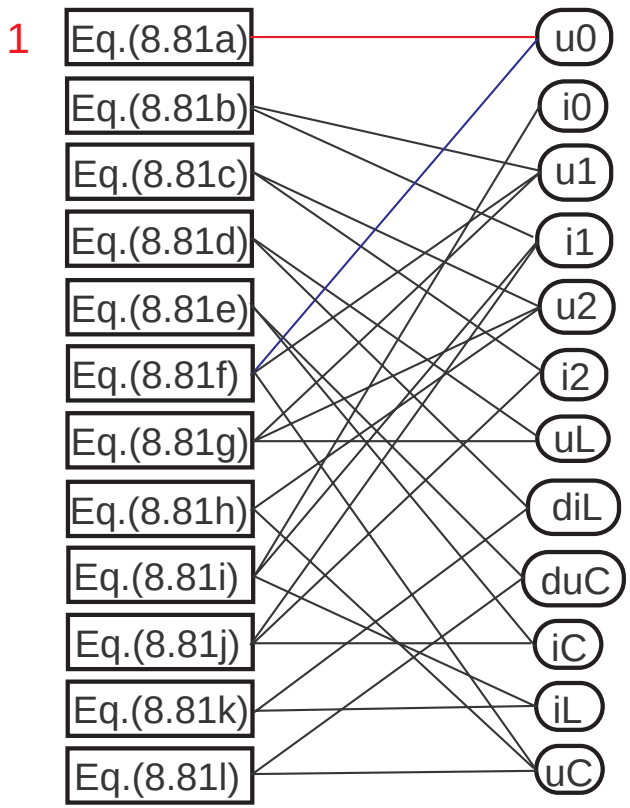


La estructura de las ecuaciones puede representarse por un *gráfico bipartito* que se llama el *dígrafo de la estructura*.

La Coloración del Dígrafo

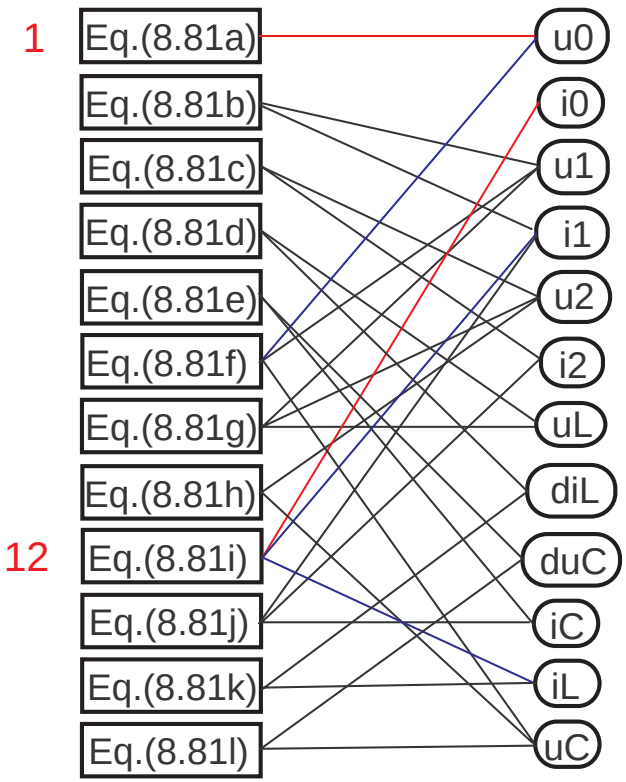


La Coloración del Dígrafo



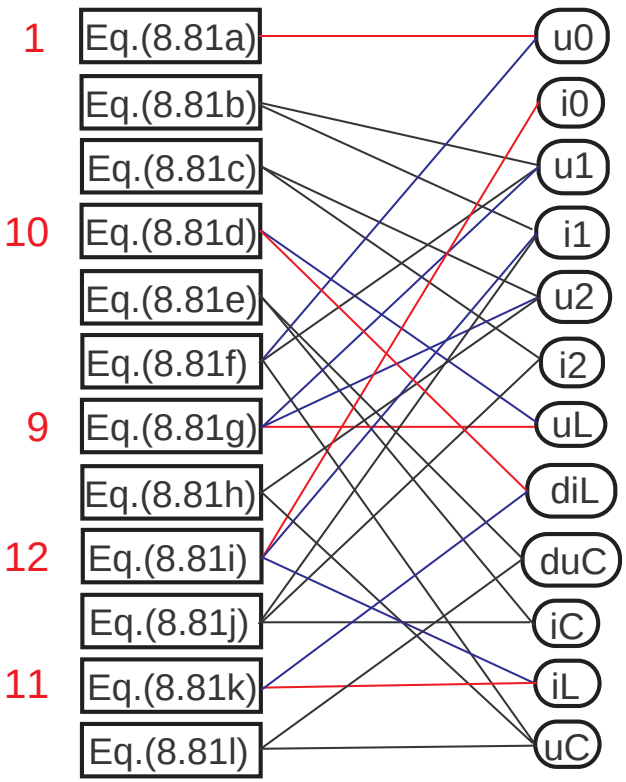
- Si una ecuación tiene solamente una conexión negra, esta conexión se colorea en rojo; la ecuación se enumera de nuevo empezando con 1; y todas las conexiones negras que terminan en la misma variable se colorean en azul.

La Coloración del Dígrafo



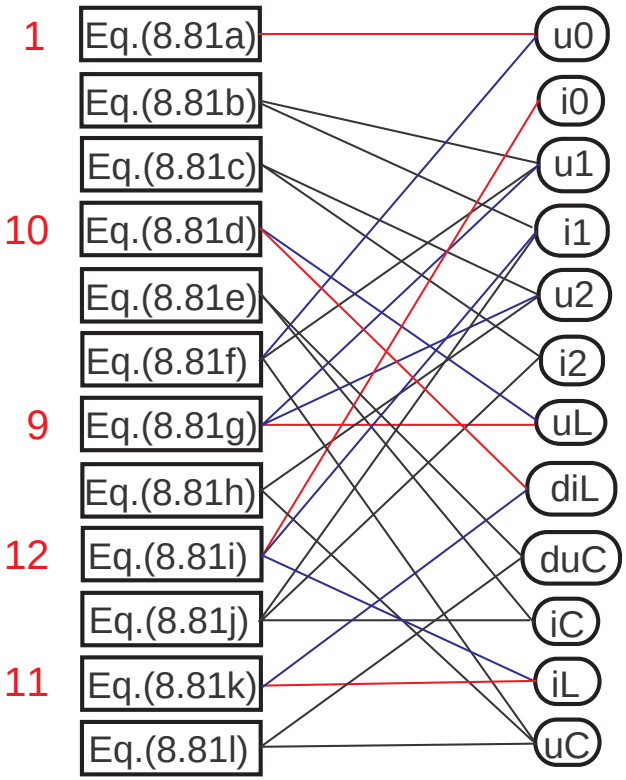
- Si una ecuación tiene solamente una conexión negra, esta conexión se colorea en rojo; la ecuación se enumera de nuevo empezando con 1; y todas las conexiones negras que terminan en la misma variable se colorean en azul.
- Si una variable tiene solamente una conexión negra, esta conexión se colorea en rojo; la ecuación se enumera de nuevo empezando con n; y todas las conexiones negras que comienzan en la misma ecuación se colorean en azul.

La Coloración del Dígrafo

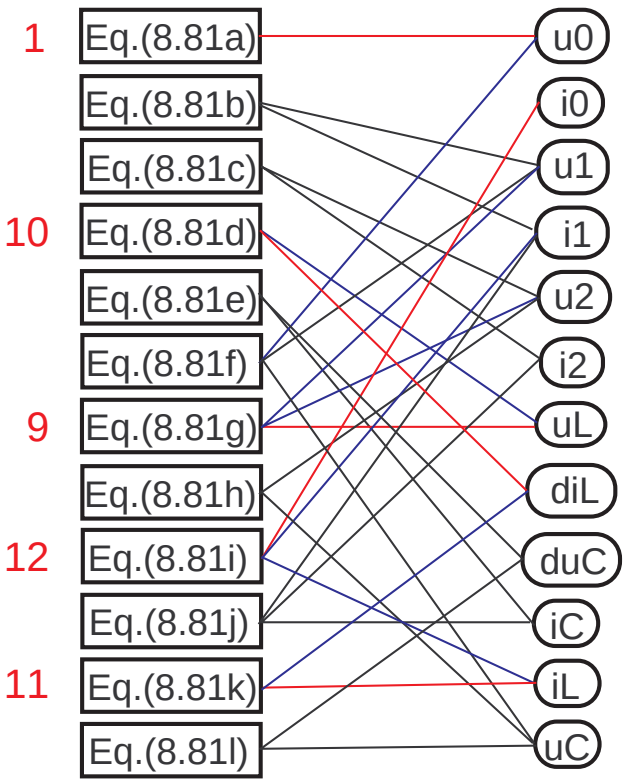


- Si una ecuación tiene solamente una conexión negra, esta conexión se colorea en rojo; la ecuación se enumera de nuevo empezando con 1; y todas las conexiones negras que terminan en la misma variable se colorean en azul.
- Si una variable tiene solamente una conexión negra, esta conexión se colorea en rojo; la ecuación se enumera de nuevo empezando con n; y todas las conexiones negras que comienzan en la misma ecuación se colorean en azul.
- Las dos instrucciones se repiten.

La Coloración del Dígrafo II

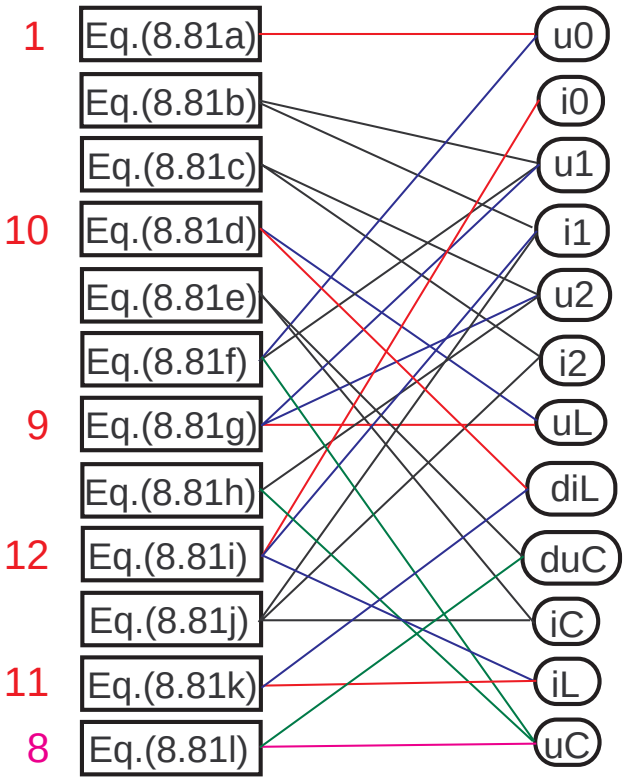


La Coloración del Dígrafo II



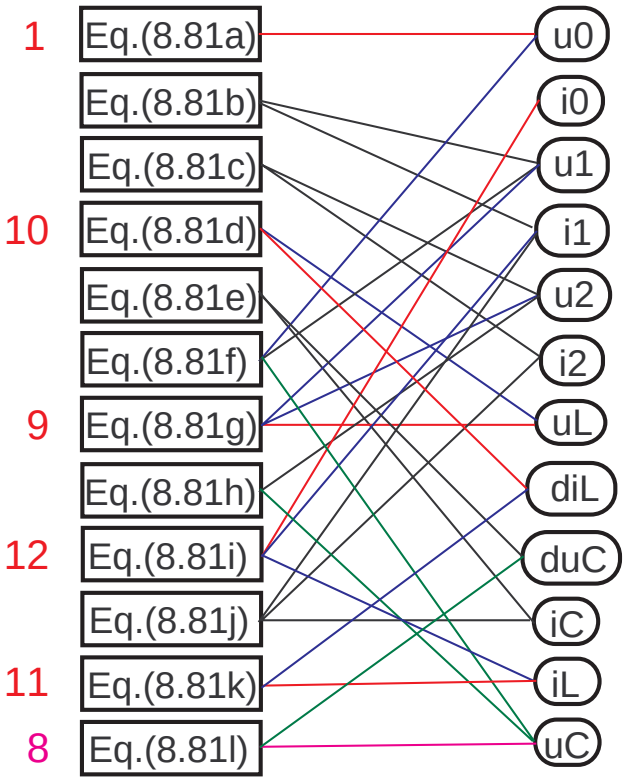
- Si llegamos a una situación donde cada ecuación aún no enumerada tiene al menos dos conexiones negras y cada variable aun no causalizada tiene al menos dos conexiones negras, se ha encontrado un *bucle algebraico*.

La Coloración del Dígrafo II

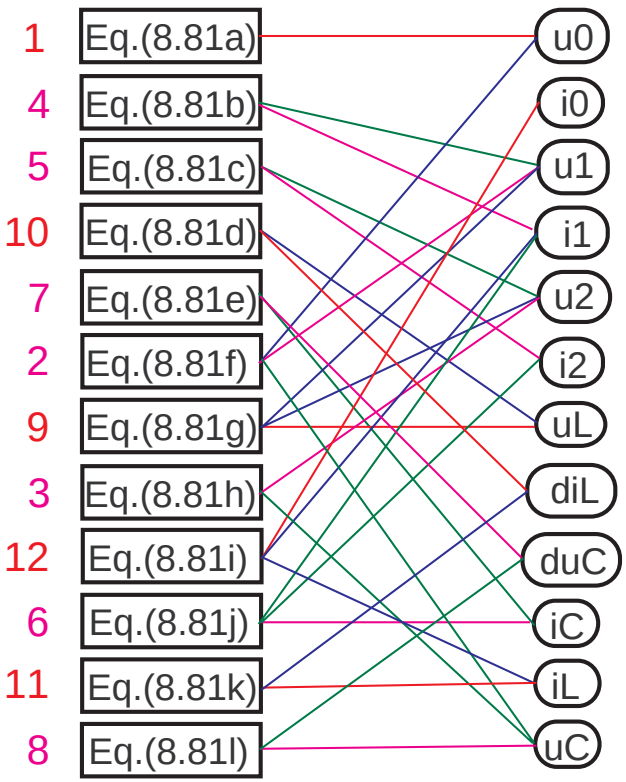


- Si llegamos a una situación donde cada ecuación aún no enumerada tiene al menos dos conexiones negras y cada variable aun no causalizada tiene al menos dos conexiones negras, se ha encontrado un *bucle algebraico*.
- Escogemos una conexión negra (existen varias heurísticas para hacerlo) y la coloreamos en rojo. La ecuación se llama *ecuación residual* y la variable se llama *variable de rasgadura*. Coloreamos todas las conexiones negras que comienzan en la ecuación residual o terminan en la variable de rasgadura en azul. La ecuación residual se enumera de nuevo usando el número más alto que aun está libre.

La Coloración del Dígrafo III

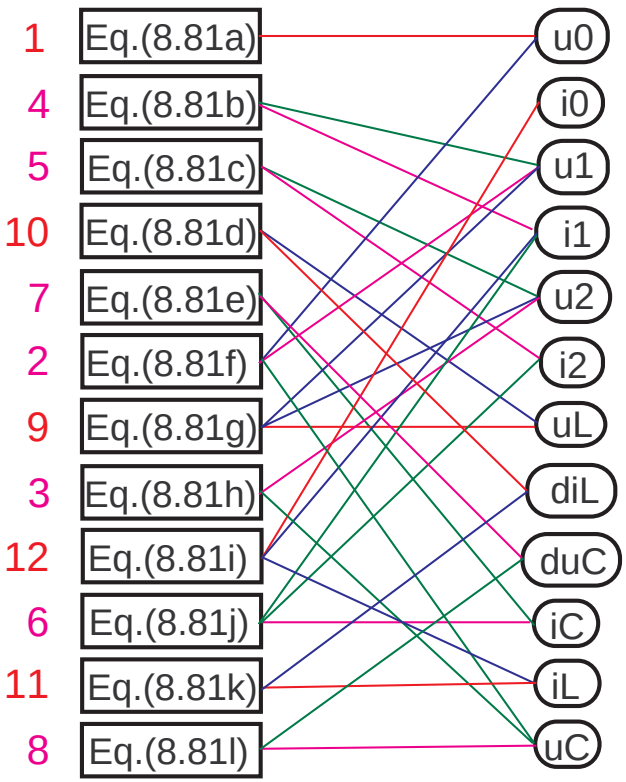


La Coloración del Dígrafo III



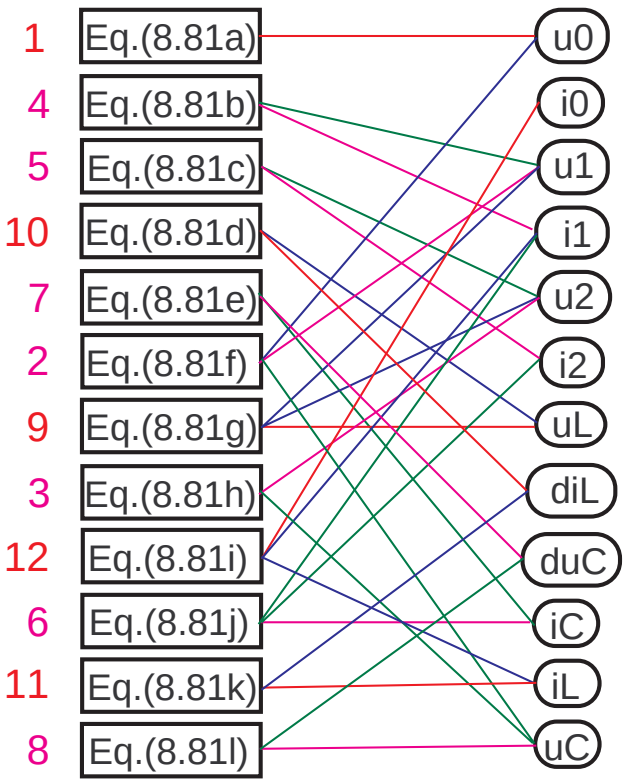
- Continuamos con las mismas instrucciones.

La Coloración del Dígrafo III



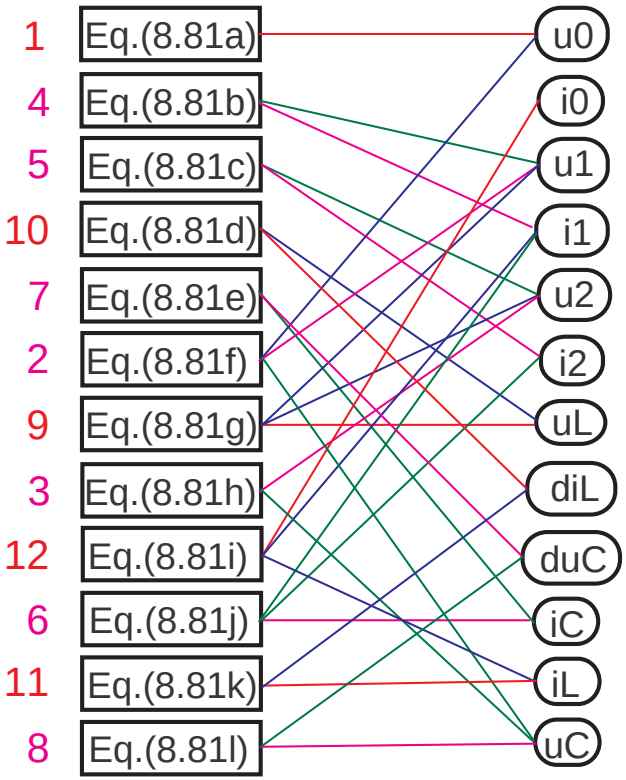
- Continuamos con las mismas instrucciones.
- Se trata de *minimizar el número de variables de rasgadura*, porque ellas servirán como variables de la iteración de Newton.

La Coloración del Dígrafo III



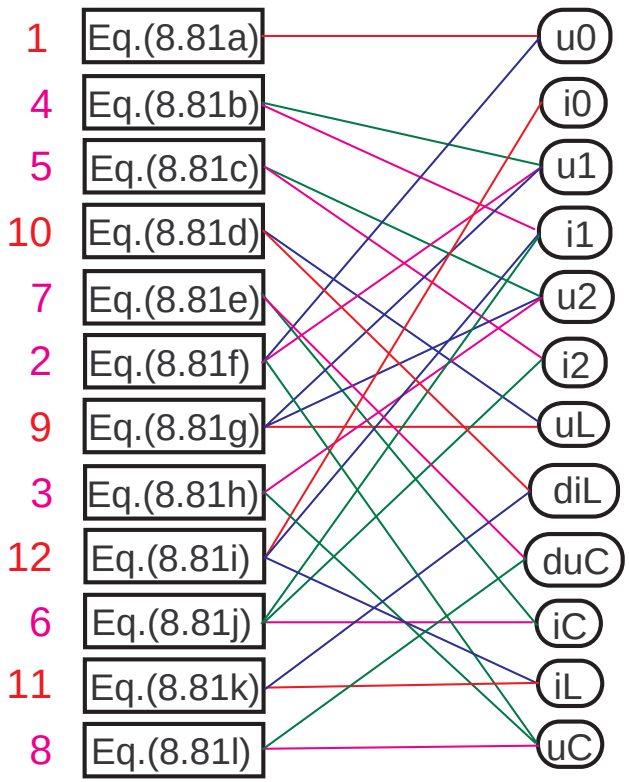
- Continuamos con las mismas instrucciones.
- Se trata de *minimizar el número de variables de rasgadura*, porque ellas servirán como variables de la iteración de Newton.
- En este ejemplo fue posible colorear el dígrafo completamente usando *una sola variable de rasgadura* a pesar de que el modelo contiene *dos variables de estado*.

La Coloración del Dígrafo III



- Continuamos con las mismas instrucciones.
- Se trata de *minimizar el número de variables de rasgadura*, porque ellas servirán como variables de la iteración de Newton.
- En este ejemplo fue posible colorear el dígrafo completamente usando *una sola variable de rasgadura* a pesar de que el modelo contiene *dos variables de estado*.
- Es entonces cierto que la *integración en línea* en este ejemplo es *más económica que DASSL*.

La Causalización de las Ecuaciones



$$
\begin{aligned}
u_0 &= f(t)\\
u_1 &= R_1 \cdot i_1\\
u_2 &= R_2 \cdot i_2\\
u_L &= L \cdot di_L\\
i_C &= C \cdot du_C\\
u_0 &= u_1 + u_C\\
u_L &= u_1 + u_2\\
u_C &= u_2\\
i_0 &= i_1 + i_L\\
i_1 &= i_2 + i_C\\
i_L &= \text{viejo}(i_L) + \bar{h} \cdot di_L\\
u_C &= \text{viejo}(u_C) + \bar{h} \cdot du_C
\end{aligned}
$$

$$
\begin{aligned}
u_0 &= f(t)\\
\hline
u_1 &= u_0 - u_C\\
u_2 &= u_C\\
i_1 &= R_1 u_1 / R_1\\
i_2 &= u_2 / R_2\\
i_C &= i_1 - i_2\\
du_C &= i_C / C\\
u_C &= \text{viejo}(u_C) + \bar{h} \cdot du_C\\
\hline
u_L &= u_1 + u_2\\
di_L &= u_L / L\\
i_L &= \text{viejo}(i_L) + \bar{h} \cdot di_L\\
i_0 &= i_1 + i_L
\end{aligned}
$$

La Integración en Línea de los Algoritmos IRK

Tratemos de aplicar la misma idea al *algoritmo Radau IIA(3)*:

$$\mathbf{x}_{\mathbf{k}+\frac{1}{3}} = \mathbf{x}_{\mathbf{k}} + \frac{5h}{12} \cdot \dot{\mathbf{x}}_{\mathbf{k}+\frac{1}{3}} - \frac{h}{12} \cdot \dot{\mathbf{x}}_{\mathbf{k}+1}$$

$$\mathbf{x}_{\mathbf{k}+1} = \mathbf{x}_{\mathbf{k}} + \frac{3h}{4} \cdot \dot{\mathbf{x}}_{\mathbf{k}+\frac{1}{3}} + \frac{h}{4} \cdot \dot{\mathbf{x}}_{\mathbf{k}+1}$$

La Integración en Línea de los Algoritmos IRK

Tratemos de aplicar la misma idea al *algoritmo Radau IIA(3)*:

$$\mathbf{x_{k+\frac{1}{3}}} = \mathbf{x_k} + \frac{5h}{12} \cdot \dot{\mathbf{x}}_{\mathbf{k+\frac{1}{3}}} - \frac{h}{12} \cdot \dot{\mathbf{x}}_{\mathbf{k+1}}$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{3h}{4} \cdot \dot{\mathbf{x}}_{\mathbf{k+\frac{1}{3}}} + \frac{h}{4} \cdot \dot{\mathbf{x}}_{\mathbf{k+1}}$$

Se trata de un algoritmo en dos etapas. La primera etapa se evalúa en el instante de tiempo $t_{k+\frac{1}{3}}$, mientras que la segunda etapa se evalúa en el instante de tiempo t_{k+1}.

La Integración en Línea de los Algoritmos IRK

Tratemos de aplicar la misma idea al *algoritmo Radau IIA(3)*:

$$\mathbf{x_{k+\frac{1}{3}}} = \mathbf{x_k} + \frac{5h}{12} \cdot \dot{\mathbf{x}}_{\mathbf{k+\frac{1}{3}}} - \frac{h}{12} \cdot \dot{\mathbf{x}}_{\mathbf{k+1}}$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{3h}{4} \cdot \dot{\mathbf{x}}_{\mathbf{k+\frac{1}{3}}} + \frac{h}{4} \cdot \dot{\mathbf{x}}_{\mathbf{k+1}}$$

Se trata de un algoritmo en dos etapas. La primera etapa se evalúa en el instante de tiempo $t_{k+\frac{1}{3}}$, mientras que la segunda etapa se evalúa en el instante de tiempo t_{k+1}.

Sin embargo, no se trata de una *predicción* seguida por una *corrección*. Hay *una sola iteración de Newton* que se extiende sobre todas las ecuaciones de las dos etapas.

La Integración en Línea de los Algoritmos IRK

Tratemos de aplicar la misma idea al *algoritmo Radau IIA(3)*:

$$\mathbf{x_{k+\frac{1}{3}}} = \mathbf{x_k} + \frac{5h}{12} \cdot \dot{\mathbf{x}}_{\mathbf{k+\frac{1}{3}}} - \frac{h}{12} \cdot \dot{\mathbf{x}}_{\mathbf{k+1}}$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{3h}{4} \cdot \dot{\mathbf{x}}_{\mathbf{k+\frac{1}{3}}} + \frac{h}{4} \cdot \dot{\mathbf{x}}_{\mathbf{k+1}}$$

Se trata de un algoritmo en dos etapas. La primera etapa se evalúa en el instante de tiempo $t_{k+\frac{1}{3}}$, mientras que la segunda etapa se evalúa en el instante de tiempo t_{k+1}.

Sin embargo, no se trata de una *predicción* seguida por una *corrección*. Hay *una sola iteración de Newton* que se extiende sobre todas las ecuaciones de las dos etapas.

Todas las ecuaciones tienen que evaluarse simultáneamente aunque las variables involucradas representen instantes diferentes en el tiempo.

La Integración en Línea de los Algoritmos IRK

Tratemos de aplicar la misma idea al *algoritmo Radau IIA(3)*:

$$\mathbf{x_{k+\frac{1}{3}}} = \mathbf{x_k} + \frac{5h}{12} \cdot \dot{\mathbf{x}}_{\mathbf{k+\frac{1}{3}}} - \frac{h}{12} \cdot \dot{\mathbf{x}}_{\mathbf{k+1}}$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{3h}{4} \cdot \dot{\mathbf{x}}_{\mathbf{k+\frac{1}{3}}} + \frac{h}{4} \cdot \dot{\mathbf{x}}_{\mathbf{k+1}}$$

Se trata de un algoritmo en dos etapas. La primera etapa se evalúa en el instante de tiempo $t_{k+\frac{1}{3}}$, mientras que la segunda etapa se evalúa en el instante de tiempo t_{k+1}.

Sin embargo, no se trata de una *predicción* seguida por una *corrección*. Hay *una sola iteración de Newton* que se extiende sobre todas las ecuaciones de las dos etapas.

Todas las ecuaciones tienen que evaluarse simultáneamente aunque las variables involucradas representen instantes diferentes en el tiempo.

Para poder hacerlo "en línea", tenemos que *duplicar las ecuaciones*.

La Integración en Línea de los Algoritmos IRK II

Usando el mismo ejemplo que antes, obtenemos:

$$v_0 = f(t + \frac{h}{3})$$
$$v_1 = R_1 \cdot j_1$$
$$v_2 = R_2 \cdot j_2$$
$$v_L = L \cdot dj_L$$
$$j_C = C \cdot dv_C$$
$$v_0 = v_1 + v_C$$
$$v_L = v_1 + v_2$$
$$v_C = v_2$$
$$j_0 = j_1 + j_L$$
$$j_1 = j_2 + j_C$$
$$j_L = \text{viejo}(i_L) + \frac{5h}{12} \cdot dj_L - \frac{h}{12} \cdot di_L$$
$$v_C = \text{viejo}(u_C) + \frac{5h}{12} \cdot dv_C - \frac{h}{12} \cdot du_C$$

$$u_0 = f(t + h)$$
$$u_1 = R_1 \cdot i_1$$
$$u_2 = R_2 \cdot i_2$$
$$u_L = L \cdot di_L$$
$$i_C = C \cdot du_C$$
$$u_0 = u_1 + u_C$$
$$u_L = u_1 + u_2$$
$$u_C = u_2$$
$$i_0 = i_1 + i_L$$
$$i_1 = i_2 + i_C$$
$$i_L = \text{viejo}(i_L) + \frac{3h}{4} \cdot dj_L + \frac{h}{4} \cdot di_L$$
$$u_C = \text{viejo}(u_C) + \frac{3h}{4} \cdot dv_C + \frac{h}{4} \cdot du_C$$

La Integración en Línea de los Algoritmos IRK III

La Integración en Línea de los Algoritmos IRK III

- En lugar de tratar con 12 ecuaciones en 12 variables, tenemos ahora 24 ecuaciones en 24 variables.

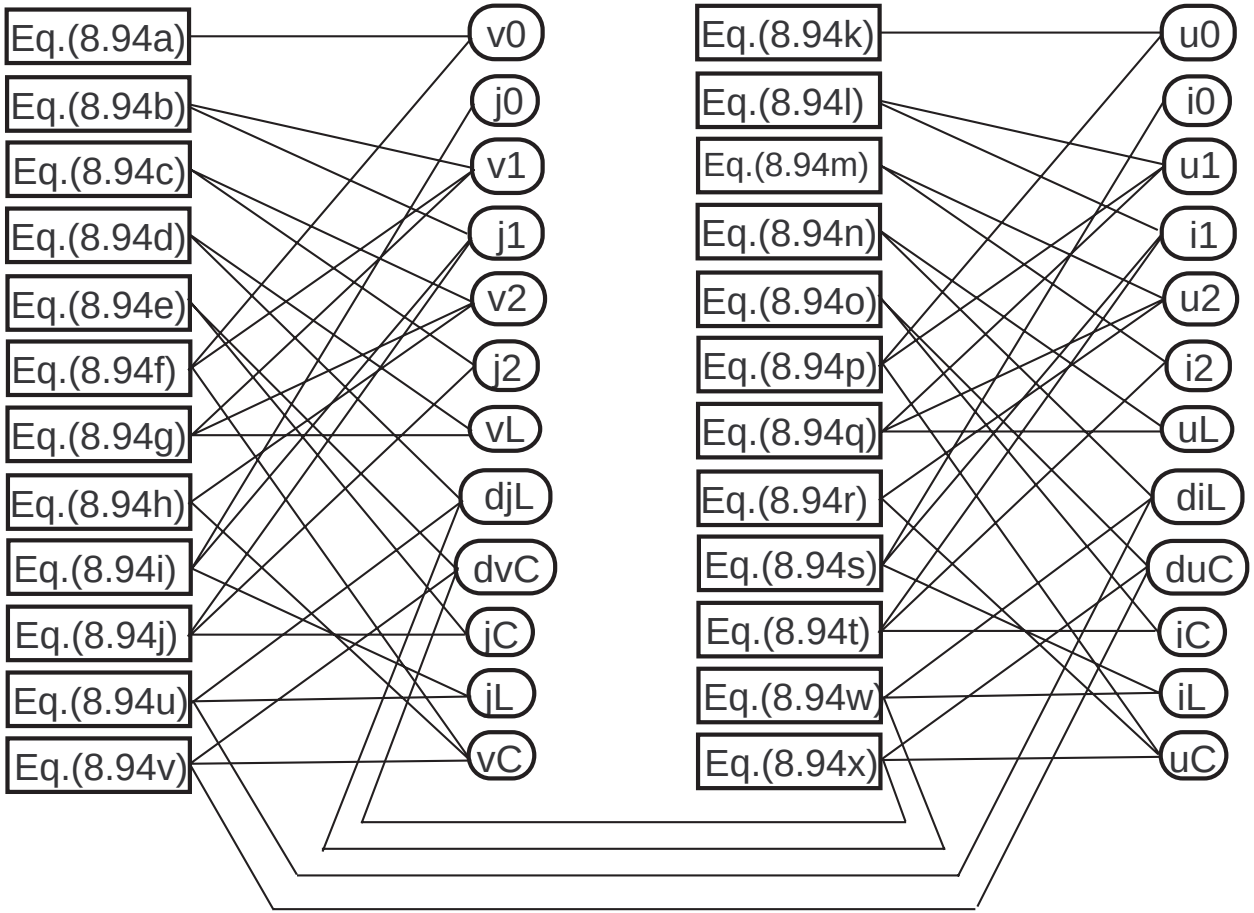
La Integración en Línea de los Algoritmos IRK III

- En lugar de tratar con 12 ecuaciones en 12 variables, tenemos ahora 24 ecuaciones en 24 variables.
- Aunque podemos utilizar pasos más grandes usando un algoritmo IRK en lugar del algoritmo BDF, cada evaluación del modelo es más cara.

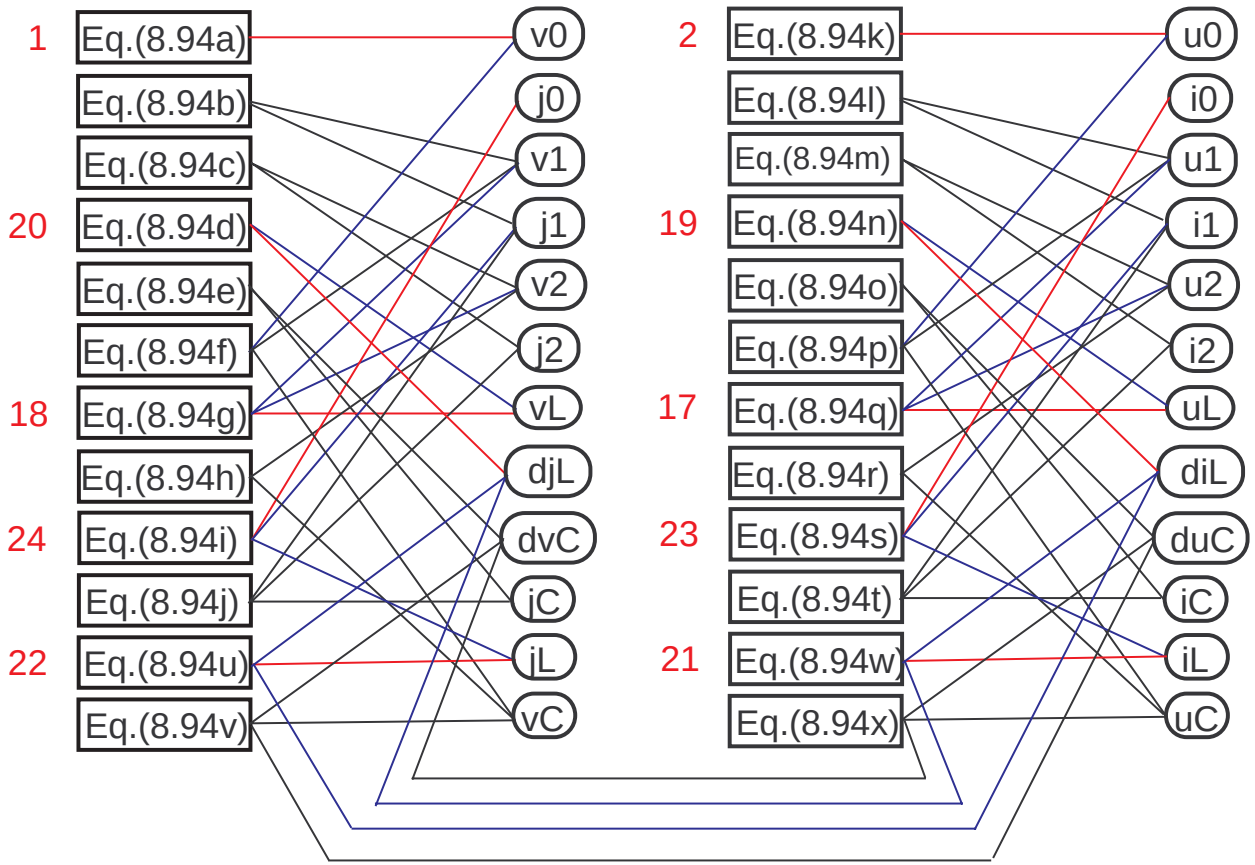
La Integración en Línea de los Algoritmos IRK III

- En lugar de tratar con 12 ecuaciones en 12 variables, tenemos ahora 24 ecuaciones en 24 variables.
- Aunque podemos utilizar pasos más grandes usando un algoritmo IRK en lugar del algoritmo BDF, cada evaluación del modelo es más cara.
- Tenemos que analizar el *número de variables de rasgadura* que resultan. Esto determinará la *economía del algoritmo*.

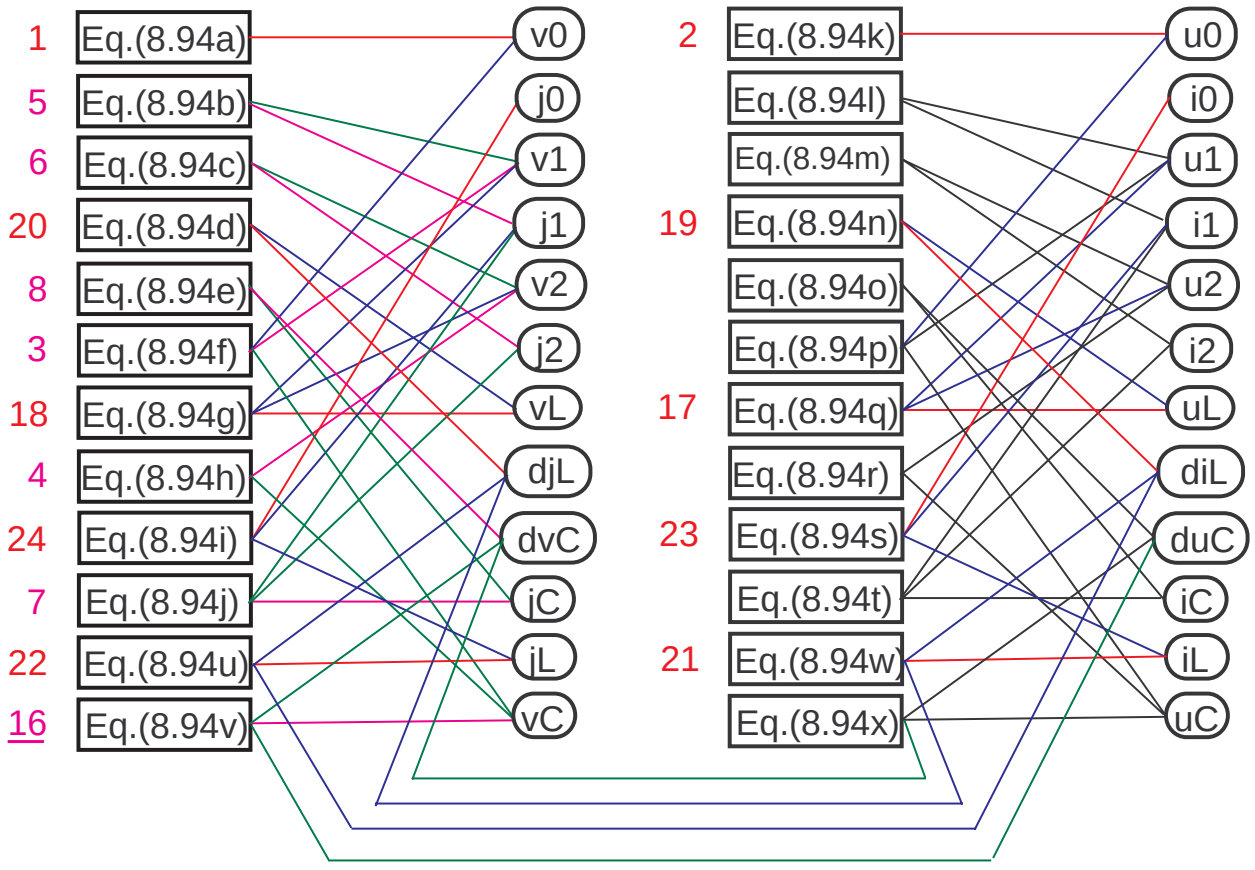
La Integración en Línea de los Algoritmos IRK IV



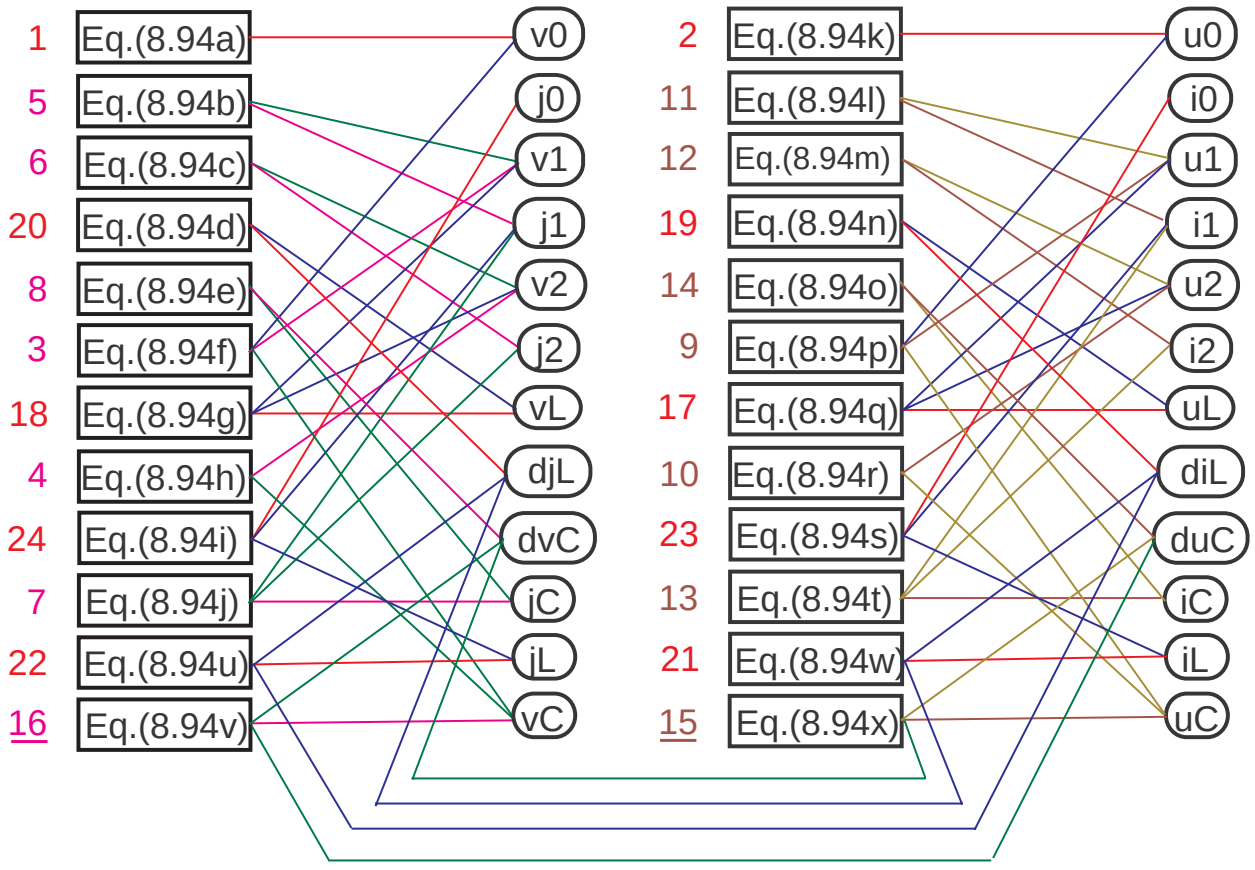
La Integración en Línea de los Algoritmos IRK IV



La Integración en Línea de los Algoritmos IRK IV



La Integración en Línea de los Algoritmos IRK IV



La Integración en Línea de los Algoritmos IRK V

La Integración en Línea de los Algoritmos IRK V

- Pudimos causalizar 10 de las 24 ecuaciones sin problemas.

La Integración en Línea de los Algoritmos IRK V

- Pudimos causalizar 10 de las 24 ecuaciones sin problemas.
- Escogimos una primera variable de rasgadura. Esto nos permitió causalizar siete ecuaciones más.

La Integración en Línea de los Algoritmos IRK V

- Pudimos causalizar 10 de las 24 ecuaciones sin problemas.
- Escogimos una primera variable de rasgadura. Esto nos permitió causalizar siete ecuaciones más.
- Tuvimos que escoger una segunda variable de rasgadura. Ahora pudimos causalizar las últimas siete ecuaciones.

La Integración en Línea de los Algoritmos IRK V

- Pudimos causalizar 10 de las 24 ecuaciones sin problemas.
- Escogimos una primera variable de rasgadura. Esto nos permitió causalizar siete ecuaciones más.
- Tuvimos que escoger una segunda variable de rasgadura. Ahora pudimos causalizar las últimas siete ecuaciones.
- En comparación con el algoritmo BDF tenemos dos veces más ecuaciones. Cada evaluación del modelo ahora necesita una iteración de Newton en 14 ecuaciones y dos variables en lugar de siete ecuaciones en una sola.

La Integración en Línea de los Algoritmos IRK V

- Pudimos causalizar 10 de las 24 ecuaciones sin problemas.
- Escogimos una primera variable de rasgadura. Esto nos permitió causalizar siete ecuaciones más.
- Tuvimos que escoger una segunda variable de rasgadura. Ahora pudimos causalizar las últimas siete ecuaciones.
- En comparación con el algoritmo BDF tenemos dos veces más ecuaciones. Cada evaluación del modelo ahora necesita una iteración de Newton en 14 ecuaciones y dos variables en lugar de siete ecuaciones en una sola.
- Sin embargo, los pasos más grandes que pueden usarse en la simulación con Radau IIA y el control del paso más barato que ofrecen los algoritmos IRK permiten predecir que la simulación con Radau IIA en línea será más económica que la simulación con BDF en línea.

Conclusiones

En esta presentación hablamos en primer lugar de DASSL, el código de la integración numérica de sistemas dinámicos más exitoso en el mercado de hoy.

Conclusiones

En esta presentación hablamos en primer lugar de DASSL, el código de la integración numérica de sistemas dinámicos más exitoso en el mercado de hoy.

Después introdujimos la integración en línea, un método que nos permite implementar algoritmos implícitos como los algoritmos del tipo BDF y los algoritmos del tipo IRK de una forma más económica.

Conclusiones

En esta presentación hablamos en primer lugar de DASSL, el código de la integración numérica de sistemas dinámicos más exitoso en el mercado de hoy.

Después introdujimos la integración en línea, un método que nos permite implementar algoritmos implícitos como los algoritmos del tipo BDF y los algoritmos del tipo IRK de una forma más económica.

Todos los métodos introducidos en esta presentación están implementados en Dymola, el entorno más avanzado de modelado y simulación de sistemas dinámicos grandes que existe hoy día.