TEMA 4:

TIPOS ESPECIALES DE PROBLEMAS ORDINARIOS DEL VALOR INICIAL EN INGENIERÍA QUÍMICA. MÉTODOS NUMÉRICOS DE RESOLUCIÓN.

Asignatura: Cálculo Avanzado de Procesos Químicos.
Titulación: Ingeniería Química
Curso: Cuarto
Cuatrimestre: Primero

1. RESOLUCIÓN NUMÉRICA DE ECUACIONES ODE-IVP DE ORDEN SUPERIOR A UNO

Las aproximaciones numéricas para la resolución de problemas ODE-IVP se basan en la evaluación de la primera derivada de una función, por lo tanto solo pueden aplicarse a ecuaciones diferenciales ordinarias de primer orden. En Ingeniería Química se presenta muchas veces la necesidad de resolver problemas ODE-IVP de orden superior a uno.

El primer paso para la resolución de una ecuación ODE-IVP de orden ***n*** es transformarla en un sistema de ***n*** ecuaciones de primer orden, se dice entonces que la ecuación está expresada en su forma canónica.

Para obtener la forma canónica de una ecuación ODE de orden n de la forma:

$$\frac{d^n z}{dt^n} = G\left(z, \frac{dz}{dt}, \frac{d^2 z}{dt^2}, ..., \frac{d^{n-1} z}{dt^{n-1}}, t\right) \text{con},$$

$$z(0) = z_0, z'(o) = z_0', ..., z^{n-1}(0) = z_0^{n-1}$$

se crean las nuevas variables mostradas en la Tabla 4.1. :

Tabla 4.1. Cambio de variable general para obtener la forma canónica de una ecuación diferencial de orden n

Término	Nueva variable
z	y_1
$\frac{dz}{dt} = \frac{dy_1}{dt}$	y_2
$\frac{d^2 z}{dt^2} = \frac{dy_2}{dt}$	y_3
----------	----------
$\frac{d^{n-1} z}{dt^{n-1}} = \frac{dy_{n-1}}{dt}$	y_n
$\frac{d^n z}{dt^n}$	$\frac{dy_n}{dt}$

Al sustituir el nuevo conjunto de variables en la ecuación de orden ***n*** se obtiene un sistema de ***n*** ecuaciones de primer orden equivalente a la ecuación inicial:

$$\begin{cases} \dfrac{dy_n}{dt} = G(y_n, y_{n-1}, \ldots\ldots, y_3, y_2, y, t) & y_n(0) = y_0^n \\ \dfrac{dy_{n-1}}{dt} = y_n & y_{n-1}(0) = y_0^{n-1} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\ \dfrac{dy_2}{dt} = y_3 & y_2(0) = y_0^2 \\ \dfrac{dy_1}{dt} = y_2 & y_1(0) = y_0^1 \end{cases}$$

Si en el lado derecho de las ecuaciones diferenciales no aparece explícitamente la variable independiente se dice que el sistema es **autónomo**. Si la variable independiente aparece explícitamente el sistema se denomina **no autónomo**.

Para resolver el sistema equivalente a la ecuación de orden ***n*** es imprescindible disponer del mismo número de condiciones iniciales que de ecuaciones en el sistema.

2. RESOLUCIÓN NUMÉRICA DE SISTEMAS DE ECUACIONES ODE-IVP

Dentro del área de Ingeniería Química, los problemas del tipo sistemas ODE-IVP representan casos en los que más de una propiedad (temperatura, concentración, etc.) varía en función de UNA sola variable (normalmente tiempo aunque también puede ser longitud).

En estos problemas se conoce:

a) cómo es la variación de las propiedades respecto a la variable: $y_n' = f(t, y_1, \ldots, y_n)\, y$

b) el valor de las propiedades en el primer valor de la variable (punto inicial del intervalo de integración): $y_n(0) = y_{n0}$.

Y se busca la función que relacione el valor de las propiedades con cada valor de la variable: $y_n = f(t)$.

Ejemplos de este tipo de problemas :

- Modelos de reactores de flujo pistón no isotermos sin dispersión.
- Reactor de lecho fijo no isotermo y sin dispersión en estado estacionario con velocidad de reacción.
- Reactores discontinuos con reacciones múltiples.

- Conducta dinámica de un sistema de control multivariable.
- Transformación de cierto tipo de problemas PDE.

La expresión general de un sistema de ecuaciones diferenciales de primer orden del valor inicial es:

$$\left.\begin{array}{l} \dfrac{dy_1}{dx} = f_1\left(x, \bar{y}\right) \\ \dfrac{dy_2}{dx} = f_2\left(x, \bar{y}\right) \\ \text{- - - - - - - -} \\ \dfrac{dy_n}{dx} = f_n\left(x, \bar{y}\right) \end{array}\right\} \Rightarrow \quad \frac{\overline{dy}}{dx} = \bar{f}\left(x, \bar{y}\right) \quad \Rightarrow \quad \bar{y}' = \bar{f}\left(x, \bar{y}\right)$$

$$\bar{y}(0) = \overline{y_0} \qquad \bar{y}(0) = \overline{y_0} \qquad \bar{y}(0) = \overline{y_0}$$

Los métodos para ecuaciones ODE-IVP descritos en el Tema 3 son aplicables a sistemas ODE-IVP haciendo las pertinentes transformaciones:

Método Euler Explícito $\Rightarrow \overline{u_{i+1}} = \overline{u_i} + h * f\left(x_i, \overline{u_i}\right)$ **con** $\overline{u_0} = \overline{y_0}$

Método Euler Implícito $\Rightarrow \overline{u_{i+1}} = \overline{u_i} + h * f\left(x_{i+1}, \overline{u_{i+1}}\right)$ **con** $\overline{u_0} = \overline{y_0}$

Método Euler Modificado $\Rightarrow \overline{u_{i+1}} = \overline{u_i} + \frac{h}{2} * \left[f\left(x_i, \overline{u_i}\right) + f\left(x_i, \overline{u_{i+1}}\right)\right]$ **con** $\overline{u_0} = \overline{y_0}$

Además de las consideraciones propias de estos métodos, descritas en el Tema 3, el trabajo con sistemas de ecuaciones implica:

1. En cada nodo $i+1$ han de aproximarse todos los valores $\overline{u_{i+1}}$ antes de pasar al siguiente nodo.
2. La obtención de las aproximaciones $\overline{u_{i+1}}$ puede implicar la resolución de sistemas de ecuaciones algebraicas lineales o no lineales en función del tipo de método y de ecuación a resolver.

2.1. PROBLEMAS ASOCIADOS A SISTEMAS ODE-IVP: RIGIDEZ

Un problema asociado a los sistemas IVP es el conocido como ***Rigidez***; aparece en aquellos sistemas en los que las variables evolucionan de forma distinta a lo largo del intervalo de integración, de forma que el tamaño de paso que permite hacer el seguimiento de una de ellas no es aceptable para otras, produciendo errores en la integración del sistema.

Ejemplo 4.1:

Consideremos el problema definido por el conjunto de reacciones: $A \Leftrightarrow B \rightarrow D$, descrito por el sistema de ecuaciones ODE:

$$\frac{dA}{dt} = -k_1 * A + k_2 * B, \qquad A(0) = 1$$
$$\frac{dB}{dt} = k_1 * A - (k_2 + k_3) * B, \qquad B(0) = 0$$

Se desea obtener la evolución de A y B para los casos:

i) $k_1 = 10, \quad k_2 = 1, \quad k_3 = 1,$

ii) $k_1 = 1000, \quad k_2 = 1, \quad k_3 = 1,$

En el primer caso se obtiene el sistema:

$$\frac{dA}{dt} = -10A + B, \qquad A(0) = 1$$
$$\frac{dB}{dt} = 10A - 2B, \qquad B(0) = 0$$

Dados unos valores definidos para las constantes de reacción, aplicando un algoritmo sencillo, por ejemplo Euler Explícito, se obtienen las aproximaciones que se muestran en la Tabla 4.2. a partir de la cual se obtiene el transcurso que se muestra en la figura 4.1.

Tabla 4.2. Resultados Ejemplo (para k_1= 10, k_2=1, k_3=1.) Euler Explícito h=0,1 s

i	t_i	A_i	B_i	i	t_i	A_i	B_i
0	0	1,0	0	80	8,0	5,65 E-5	5,14 E-4
10	1,0	4,19 E-2	3,81 (-1)	90	9,0	2,20 E-5	2,00 E-4
20	2,0	1,60 E-2	1,5 E-1	100	10,0	8,55 E-6	7,78 E-5
30	3,0	6,34 E-3	5,77 E-2	11	11,0	3,33 E-6	3,03 E-5
40	4,0	2,47 E-3	2,25 E-2	120	12,0	1,29 E-6	1,18 E-5
50	5,0	9,60 E-4	8,73 E-3	130	13,0	5,03 E-7	4,58 E-6
60	6,0	3,73 E-4	3,40 E-3	140	14,0	1,96 E-7	1,78 E-6
70	7,0	1,45 E-4	1,32 E-3	150	15,0	7,62 E-8	6,93 E-7

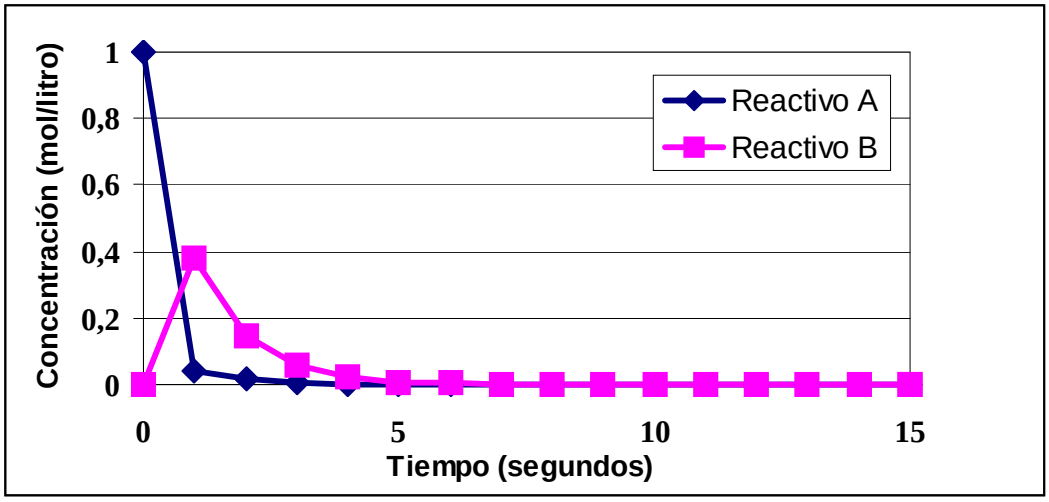

Figura 4.1. Transcurso de A y B para k_1= 10, k_2=1, k_3=1.

En el segundo caso se obtiene el sistema:

$$\frac{dA}{dt} = -1000\,A + B, \qquad A(0) = 1$$

$$\frac{dB}{dt} = 1000\,A - 2B, \qquad B(0) = 0$$

Aplicando un algoritmo Euler Explícito no se puede resolver este problema. Utilizando la subrutina LSODE, que aplica el método GEAR se obtienen las aproximaciones que se muestran en la Tabla 4.3. a partir de la cual se obtiene el transcurso de la figura 4.2.

Tabla 4.3. Resultados Ejemplo (para k_1= 1000, k_2=1, k_3=1.) LSODE h=0,2 s

i	t_i (s)	A_i	B_i	i	t_i	A_i	B_i
0	0	1	0	16	3,2	4,09 E-5	4,09 E-2
2	0,4	6,71 E-4	6,70 E-1	18	3,6	2,70 E-5	2,74 E-2
4	0,8	4,50 E-4	4,50 E-1	20	4,0	1,86 E-5	1,84 E-2
6	1,2	3,20 E-4	3,02 E-1	22	4,4	1,23 E-5	1,23 E-2
8	1,6	2,02 E-4	2,02 E-1	24	4,8	8,28 E-6	8,27 E-3
10	2,0	1,36 E-4	1,36 E-1	26	5,2	5,55 E-6	5,5 E-3
12	2,4	9,10 E-5	9,09 E-2	28	5,6	3,72 E-6	3,72 E-3
14	2,8	6,10 E-5	6,10 E-2	30	6,0	2,5 E-6	2,5 E-3

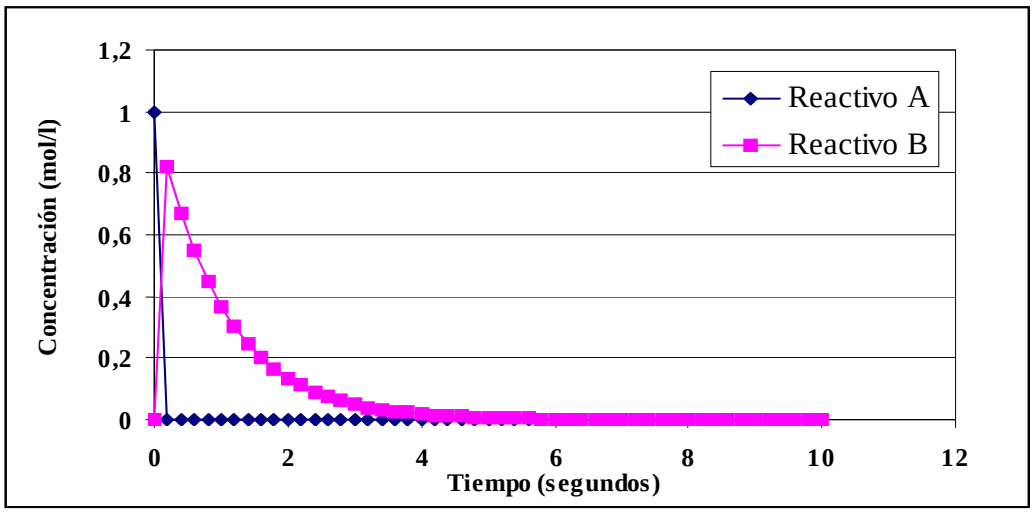

Figura 4.2. Transcurso de A y B para k_1= 1000, k_2=1, K_3=1 Con tf=10 s.

En la figura 4.2. se observa que en el intervalo de integración en el que la evolución de A ha llegado a su valor final la evolución de B no ha hecho más que empezar, es decir, el tamaño de paso de integración así como el tiempo final de integración que permiten el seguimiento de la evolución de A no parecen adecuados para conocer la evolución de B. Si quisiéramos conocer la evolución completa necesitaríamos una integración como la que muestra la Tabla 4.4 y la figura 4.3.

Tabla 4.4. Resultados Ejemplo (para k_1= 1000, k_2=1, k_3=1.) LSODE h=0,001s

i	t_i	A_i	B_i	i	t_i	A_i	B_i
0	0	1	0	16	1,6 E-2	9,86 E-4	9,84 E-1
2	2,0 E-3	1,36 E-1	8,63 E-1	18	1,8 E-2	9,83 E-4	9,83 E-1
4	4,0 E-3	1,92 E-2	9,47 E-1	20	2,0 E-2	9,81 E-4	9,82 E-1
6	6,0 E-3	3,46 E-3	9,92 E-1	22	2,2 E-2	9,79 E-4	9,80 E-1
8	8,0 E-3	9,92 E-3	9,92 E-1	24	2,4 E-2	9,77 E-4	9,79 E-1
10	1,0 E-2	1,04 E-3	9,90 E-1	26	2,6 E-2	9,75 E-4	9,78 E-1
12	1,2 E-2	9,96 E-4	9,89 E-1	28	2,8 E-2	9,73 E-4	
14	1,4 E-2	9,88 E-4	9,86 E-1	30	3,0 E-2	9,72 E-4	

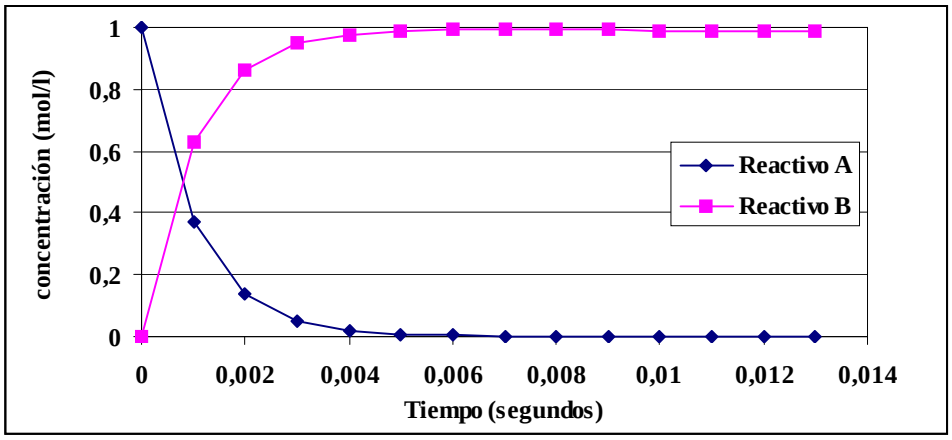

Figura 4.3. Transcurso de A y B para k_1= 1000, k_2=1, k_3=1 Con tf=0,014 s

Con las condiciones de integración utilizadas en la Figura 4.2 la evolución de B quedaba perfectamente reflejada pero no podía decirse lo mismo de la evolución de A.

El problema i) se clasifica como ***NO RÍGIDO*** mientras que el problema correspondiente al caso ii) se clasifica como ***RÍGIDO***.

Las soluciones exactas de los sistemas de ecuaciones lineales vienen dadas por los valores propios del determinante de la matriz de coeficientes del sistema. La relación entre los valores propios máximo y mínimo de un sistema de ecuaciones se cuantifica mediante la **RELACIÓN DE RIGIDEZ (RR)**:

$$RR = \frac{\text{máx}_i \left|\text{parte real de } \lambda\right|}{\text{mín}_i \left|\text{parte real de } \lambda\right|}, \qquad i = 1, 2, \ldots\ldots\ldots, n \text{ donde } (n = n^o \text{ de ecuaciones})$$

No existe un valor límite de RR concreto para definir a un sistema como Rígido o no Rígido, algunos autores (Davis, 1990) consideran que valores de RR alrededor de 20 (o menores) son propios de sistemas no rígidos, valores de RR cercanos a 1000 denotan sistemas con cierta rigidez y valores de RR iguales o superiores a 10^6 son propios de sistemas muy rígidos.

Ejemplo 4.2:

En el problema anterior, para el caso i, tenemos una matriz de coeficientes de la forma: $\begin{bmatrix} -10 & 1 \\ 10 & -2 \end{bmatrix}$.

Los valores propios (autovalores) del sistema son un conjunto de valores que hacen que se cumpla que:

$(A - \lambda I)K = 0$ siendo:

A= matriz de coeficientes.
I= matriz identidad
K= vector propio del sistema
λ = valores propios del sistema
$(A - \lambda I)$= ecuación característica del sistema.

Entonces para obtener los valores propios: $(A - \lambda I) = 0$, es decir:

$$\begin{vmatrix} -10-\lambda & 1 \\ 10 & -2-\lambda \end{vmatrix} = 0 \Rightarrow \lambda_1 = -0{,}9, \quad \lambda_2 = -11 \Rightarrow RR = \frac{11}{0{,}9} \approx 12 \Rightarrow \text{NO RIGIDO}$$

para el caso ii), tenemos una matriz de coeficientes de la forma: $\begin{bmatrix} -1000 & 1 \\ 1000 & -2 \end{bmatrix}$.

$$\Rightarrow \begin{vmatrix} -1000-\lambda & 1 \\ 1000 & -2-\lambda \end{vmatrix} = 0 \Rightarrow \lambda_1 = -1, \quad \lambda_2 = -997 \Rightarrow RR = \frac{997}{1} \approx 997 \Rightarrow \text{RIGIDO}$$

Si el sistema de ecuaciones es no lineal, se linealiza en cada nodo, obteniéndose:

$$\frac{d\bar{y}}{dt} = \bar{\bar{Q}}(t_i) * \bar{y}(t_i) + \bar{J}(t_i)\left(\bar{y} - \bar{y}(t_i)\right), \text{ donde: } \begin{array}{l} \bar{\bar{Q}}(t_i) = \text{matriz de coeficientes evaluada a tiempo } t_i \\ \bar{y}(t_i) = \text{vector y evaluado a tiempo } t_i \\ \bar{J}(t_i) = \text{matriz Jacobiana evaluada a tiempo } t_i \end{array}$$

Para problemas no lineales la RIGIDEZ está basada en los valores propios de la matriz jacobiana y por lo tanto se aplica sólo a un nodo específico pudiendo variar a lo largo del intervalo de integración.

El problema que plantean los sistemas rígidos (y especialmente los problemas no lineales) es que necesitan técnicas que sean capaces de actuar bien sobre el problema. En general los métodos explícitos no son aplicables a problemas rígidos, además suele ser necesario utilizar algoritmos que permitan cambiar el tamaño de paso y el orden de integración a lo largo del problema.

3. ALGORITMOS COMERCIALES PARA PROBLEMAS ODE-IVP

3.1. CARACTERÍSTICAS DE LOS ALGORITMOS DE ORDEN SUPERIOR

Los algoritmos mostrados hasta ahora no contemplan la posibilidad de cambios de tamaño de paso, de cambio de orden del algoritmo o de cambios de la tolerancia establecida de acuerdo a las necesidades del problema a lo largo de la integración. Sin embargo, la integración de sistemas complejos, con gran número de variables y comportamientos rígidos, sería imposible con algoritmos tan estrictos.

Los programas de cálculo comerciales contienen algoritmos más sofisticados capaces de controlar el error local de corte a lo largo de todos los pasos o de mejorar la eficiencia a lo largo de un problema rígido. Para ello deben ser capaces de cambiar el tamaño de paso a lo largo del problema. Cuando se trabaja con sistemas de ecuaciones se establece una tolerancia para cada ecuación y el criterio para pasar al siguiente nodo con el mismo tamaño de paso es que TODAS las ecuaciones cumplan su criterio de tolerancia.

Ejemplo 4.3:

Si, por ejemplo, el error cometido al usar un tamaño de paso con un método de orden P es $e = \Phi\left(h_1^{p+1}\right)$ y se desea que el error no supere un valor de tolerancia definido por $TOL = \Phi\left(h_2^{p+1}\right)$, si e> TOL se determina que la variación del tamaño de paso debe ser tal que $h_2 = h_1 * \left[\frac{TOL}{e}\right]^{\frac{1}{(p+1)}}$, para evitar errores se puede utilizar un h_2 que sea un porcentaje menor que el calculado mediante esta fórmula.

En los problemas rígidos el procedimiento de cálculo puede llevar a tamaños de paso muy pequeños en la zona donde la solución cambia muy rápidamente e incluso a necesitar algoritmos de alto orden de exactitud (normalmente al principio de la integración) pero tan pronto como la componente rígida se aleja, el error desciende rápidamente y sería conveniente reducir el orden e incrementar el tamaño de paso para que el coste de la integración no sea excesivo. Así mismo, las necesidades del usuario que adquiere un programa para la resolución de ODE-IVP pueden variar en cada problema; por tanto, el programa debe ser capaz de seleccionar los algoritmos más adecuados para cada tipo de problema de forma que el coste de computación no sea mayor del necesario en cada caso. Uno de los algoritmos más utilizados (recogido en la mayor parte de las librerías matemáticas) es el desarrollado por el profesor G. W. Gear, denominado ***método GEAR***, que permite variar el orden, el tamaño de paso e incluso la aproximación utilizada a lo largo de un problema. La Figura 4.4. resume esquemáticamente la forma de trabajo de este método que ha sido continuamente mejorado en diferentes subrutinas (LSODE; EPISODE, EPISODEB, etc.)

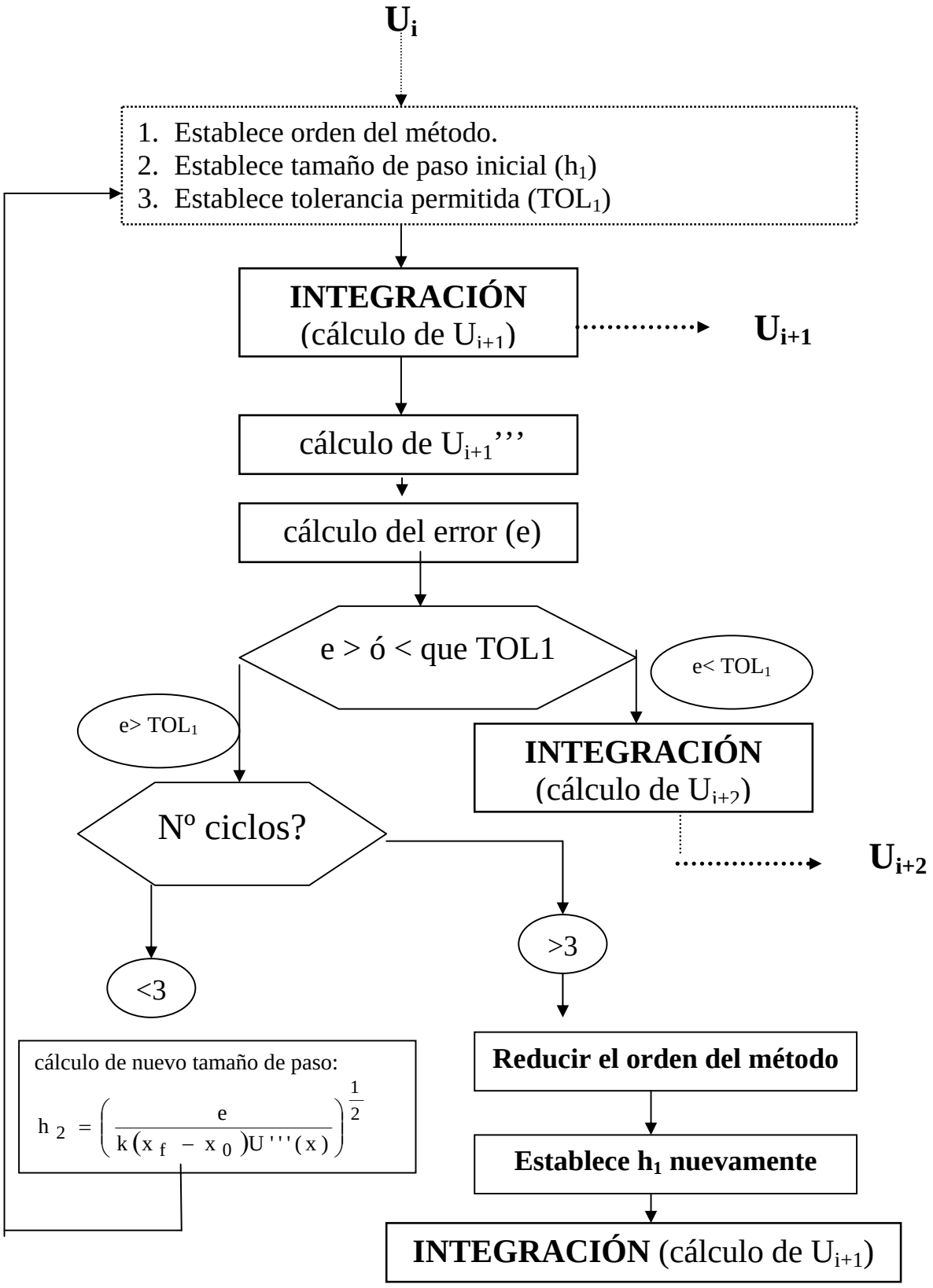

Figura 4.4. Resumen del método GEAR

3.2. LIBRERÍAS COMERCIALES

Los paquetes de programación de computadoras, programas que se encuentran en bibliotecas en la forma de subrutinas accesibles al usuario del programa principal, constituyen la forma más habitual para la resolución de problemas ODE-IVP complicados. Una subrutina para resolver ODE-IVPs será diseñada para el cómputo de una solución numérica sobre $[x_i, x_{i+1}]$ y devolver el valor de u_{i+1} dado x_i, x_{i+1} y u_i. Una típica secuencia de llamada podría ser (de forma muy simplificada):

CALL DRIVE (FUNC, X, XEND, U, TOL)

Donde:

DRIVE	= nombre de la subrutina que contiene el algoritmo IVP
FUNC	= subrutina escrita por el usuario para la evaluación de f(x,y)
X	= x_i
XEND	= x_{i+1}
U	= entrada que contiene u_i y salida que contiene u_{i+1}
TOL	= tolerancia al error.

La creación de un paquete de programas es un problema de cierta magnitud. Además una vez que se ha completado el paquete (código) este debe contener suficiente documentación para que pueda utilizarse por los usuarios (que normalmente no serán especialistas matemáticos sino profesionales de otras áreas). Algunos aspectos de la documentación son significativos:

1. Comentarios en el código que identifiquen los argumentos y proporcionen instrucciones generales al usuario.

2. Documentos con ejemplos que muestren cómo usar el código e ilustren al usuario sobre algunos aspectos prácticos del código.

3. Ejemplos sustanciales sobre el comportamiento del código en una amplia gama de problemas.

4. Ejemplos que muestren los posibles errores y las advertencias del código para reconocerlos.

Las librerías comerciales de mayor uso son:

Librería IMSL: International Mathematics and Statistics Libraries.

Librería HSL: Harwell Subroutine Library.

Librería NAG: Numerical Algorithms Group.

IMSL es un acrónimo de ***International Mathematical and Statistical Library***. Para poder linkar y utilizar las subrutinas IMSL en un ordenador es necesario pagar el coste de la licencia de uso.

IMSL contiene más de 1000 subrutinas implementadas en FORTRAN que permiten llevar a cabo los cálculos numéricos y estadísticos más frecuentes en problemas ingenieriles. Además, como se indica en el Cuadro 4.1. existen versiones de la librería IMSL implementadas también en C, C++ o Java:

Cuadro 4.1. Versiones de la Librería IMSL

1. IMSL FORTRAN Numerical Library Version 7.0. (Incluye todos los algoritmos de IMSL FORTRAN 77 Library y de IMSL FORTRAN 90 Library
2. IMSL C Numerical Library (CNL)
3. IMSL Numerical Library for Java (JNL)
4. IMSL C# Numerical Library for Microsoft® .NET Applications

Fuente: http://www.roguewave.com/products/imsl-numerical-libraries.aspx

La librería IMSL contiene una distribución general en diferentes secciones:

MATH LIBRARY ⇒ contiene algoritmos para diferentes problemas matemáticos

STAT LIBRARY ⇒ contiene algoritmos para problemas estadísticos.

MATH SPECIAL FUNCTION LIBRARY ⇒ contiene algoritmos para evaluar funciones matemáticas especiales (funciones elementales, trigonométricas, hiperbólicas, elípticas, gamma, Bessel, Mathieu, etc.)

El contenido de la IMSL MATH Library se muestra en el Cuadro 4.2.:

Cuadro 4.2. Índice de contenidos de la librería matemática IMSL

Chapter 1: Linear Systems
Chapter 2: Eigensystem Analysis
Chapter 3: Interpolation and Approximation
Chapter 4: Integration and Differentiation
Chapter 5: Differential Equations
Chapter 6: Transforms
Chapter 7: Nonlinear Equations
Chapter 8: Optimization
Chapter 9: Basic Matrix/Vector Operations
Chapter 10: Linear Algebra Operators and Generic Functions
Chapter 11: Utilities

El capítulo 5 es el dedicado a la resolución numérica de ecuaciones diferenciales. Se presentan tres subrutinas para la resolución de problemas ODE-IVP:

1. Subrutina IVPRK: Aplica métodos de Runge Kutta.
2. Subrutina IVMRK: Aplica método de Runge Kutta permitiendo variación del orden del método
3. Subrutina IVPAG: Aplica método de Adams o Gear.

HSL (inicialmente Harwell Subroutine Library) es una colección de paquetes de FORTRAN para la computación a gran escala en el área científica, escritas y desarrolladas en primer término por el Grupo de Análisis Numérico en el Laboratorio Rutherford Appleton. La librería nació en 1963 para uso de los laboratorios HARWELL.

HSL se divide en dos partes:

a) HSL 2011 $\Rightarrow$ contiene todos los paquetes de subrutinas para la resolución de problemas matemáticos en su última versión actualizada.

   HSL 2011 es un producto comercial pero también está disponible de forma gratuita para los centros académicos británicos siempre que su uso se limite a fines académicos y de investigación.

b) HSL archive ⇒ contiene paquetes de subrutinas que formaron parte de antiguas versiones de HSL.

El paquete HSL archive puede adquirirse gratuitamente por cualquier usuario siempre y cuando no se usen con intereses comerciales.: http://www.hsl.rl.ac.uk/archive/

Ambas colecciones están escritas en FORTAN 77 y FORTRAN 90; el usuario debe ser capaz de linkar las subrutinas en este lenguaje de programación.

La librería HSL 2002 comprende los paquetes de software que se muestran en el Cuadro 4.3.:

Cuadro 4.3. Paquetes de software de la librería matemática HSL 2002 (disponibles gratuitamente para uso personal en: http://www.hsl.rl.ac.uk/archive/index.html)

PAQUETE A	COMPUTER ALGEBRA
PAQUETE D	***DIFFERENTIAL EQUATIONS***
PAQUETE E	EIGENVALUES AND EIGENVECTORS
PAQUETE F	MATHEMATICAL FUNCTIONS
PAQUETE G	GEOMETRICAL PROBLEMS
PAQUETE I	INTEGER VALUED FUNCTIONS
PAQUETE K	SORTING
PAQUETE L	LINEAR PROGRAMMING
PAQUETE M	LINEAR ALGEBRA
PAQUETE N	NONLINEAR EQUATIONS
PAQUETE O	INPUT AND OUTPUT AIDS
PAQUETE P	POLYNOMIAL AND RATIONAL FUNCTIONS
PAQUETE Q	NUMERICAL INTEGRATION
PAQUETE S	STATISTICS
PAQUETE T	INTERPOLACION AND APPROXIMATION
PAQUETE V	OPTIMIZATION AND NONLINEAR DATA FITTING
PAQUETE Y	TEST PROGRAM GENERATORS
PAQUETE Z	FORTAN SYSTEM FACILITIES

El paquete D contiene las subrutinas de cálculo de Ecuaciones diferenciales ODE-IVP siguientes:

DA Runge-Kutta methods for ordinary differential equation initial value problems

DC Lineal multi-step methods, predictor corrector methods for ordinary differential equation initial value problems.

- DC03 Ordinary differential equations: Gears' method, sparse Jacobian
- DC04 Simplified calling sequence for DC03
- DC05 Ordinary differential equations: Gears' method, reverse communication
- DC06 Ordinary differential equations: advances DC05 solution forward
- DC07 Ordinary differential equations: Gears' method, full Jacobian

NAG es el acrónimo de Numerical Algorithms Group, organismo que ha desarrollado una de las librerías más potentes en el cálculo numérico: **la NAG Fortran Library**, una colección de más de 1000 subrutinas para el cálculo matemático y estadístico. La librería puede utilizarse en aplicaciones que incluyan programas en Visual Basic, VBA, Excel así como Fortran y C/C++.

Al igual que en los casos anteriores, el usuario debe ser capaz de linkar las subrutinas a su propio programa en fortran. La librería consta de las secciones que se muestran en el Cuadro 4.4. La sección D02 contiene las subrutinas destinadas a la resolución de Ecuaciones diferenciales ordinarias, contiene 62 subrutinas de las cuales se señalan a continuación las más utilizadas para problemas ODE-IVP:

D02BGF: ODEs, IVP, Runge-Kutta-Merson method, until a components attains a given value

D02BHF: ODEs, IVP, Runge-Kutta-Merson method, until function solution is zero.

D02CJF: ODEs, IVP, Adams method, until function solution is zero.

D02EJF: ODEs, stiff IVP, BDF method, until function solution is zero.

D02LAF: Second order ODEs IVP, Runge-Kutta-Nystrom Method.

D02LXF: Second order ODEs, IVP, set up for D02LAF

D02LZF: Second order ODEs, Ivp, diagnostics for D02LAF.

D02NBF: Explicit ODEs, Stiff IVP, full Jacobian

D02NCF: Explicit ODEs, stiff IVP, banded jacobian.

D02NDF: Explicit ODEs, stiff IVP, sparse jacobian

Cuadro 4.4. Contenidos de la Librería matemática NAG

Chapter A02	Complex arithmetic.
Chapter C02	Zeros of polinomials
Chapter C05	Roots of one or more transcendental equations.
Chapter C06	Summation of Series.
Chapter D01	Quadrature
Chapter D02	***Ordinary Differential Equations***
Chapter D03	***Partial Differential Equations***
Chapter D04	Numerical Differentiation
Chapter D05	Integral Equations
Chapter E02	Curve and surface Fitting
Chapter E04	Minimising or Maximising a function
Chapter F01	Matrix Factorisations.
Chapter F02	Eigenvalues and Eignvectors.
Chapter F03	Determinants
Chapter F04	Simultaneous Linear Equations.
Chapter F05	Orthogonalisation
Chapter F06	Linear Algebra Support Routines.
Chapter F07	Linear Equations (LAPACK)
Chapter F08	Least-squares and eigenvalues Problems (LAPACK)
Chapter F11	Sparse Lineal Algebra
Chapter G01	Simple calculations and statistical data
Chapter G02	Correlation and Regression analysis.
Chapter G03	Multivariate Methods.
Chapter G04	Analysis of Variance
Chapter G05	Random Number Generators.
Chapter G07	Univariate Estimation
Chapter G08	Non-parametric statistics
Chapter G10	Smoothing in Statistics
Chapter G11	Contingency table Analysis
Chapter G12	Survival Analysis
Chapter G13	Time Series Analysis.
Chapter H	Operation Research
Chapter M01	sorting
Chapter P01	Error Trapping
Chapter S	Approximations of special Functions.
Chapter X01	Mathematical constants
Chapter X02	Machine Constants
Chapter X03	Inner products.
Chapter X04	Input/Output Utilities
Chapter X05	Date and Time Utilities

4. BIBLIOGRAFÍA RELACIONADA.

Textos que desarrollan los métodos numéricos presentados a nivel de usuario con ejemplos de ingeniería química:

Davis, M. E.; *Métodos y Modelos Numéricos para Ingenieros Químicos.* CAPÍTULO 1. Compañía Editorial Continental de C.V. México, México D.F. 1990.

Riggs, J.B.; *An Introduction to Numerical Methods for Chemical Engineers* CAPÍTULO 4. Texas Tech University Press, Lubbock, Texas. 1994.

Textos que desarrollan los métodos numéricos presentados a nivel de usuario con ejemplos generales:

Gerald, C. F., Wheatley, P. O.; *Applied Numerical Analysis (5º Edición).* CAPÍTULO 5. Addison-Wesley Publishing Company. 1994.

Textos para profundizar en los métodos numéricos para ODE-IVP:

Gear, G. W.; *Numerical Initial Values Problems in Ordinary Differential Equations*. Prentice Hall INC. 1971.

Butcher, J.C. *Numerical Methods for Ordinary Differential Equations*. Wiley, Chichester, UK. 2003.

Griffiths, D.F., Higham, D.J. *Numerical Methods for Ordinary Differential Equations*. Springer, Berlín, Alemania. 2010.

Paginas Web de interés para usuarios de subrutinas comerciales:

http://www.nag.co.uk: Página web de Numerical Algorithmics Group, empresa que desarrolla las librerías Nag.

http://www.hsl.rl.ac.uk/: Página web de la librería matemática HSL

http://www.roguewave.com/products/imsl-numerical-libraries.aspx: Página web de las librerías matemáticas IMSL.