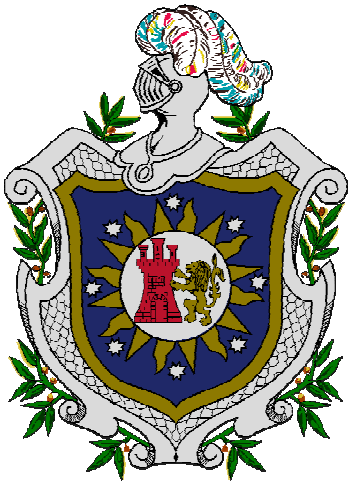
UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA- LEÓN
FACULTAD DE CIENCIAS Y TECNOLOGIA
DEPARTAMENTO DE MATEMÁTICA Y ESTADÍSTICA



APLICACIONES DE MÉTODOS NUMÉRICOS CON MATLAB

MONOGRAFÍA

PARA OPTAR AL TÍTULO DE LICENCIADO EN MATEMÁTICA

PRESENTADO POR: JORGE ABILIO PEREZ FRANK

TUTORA: MSC. ANGELA ALTAMIRANO SLINGER

LEÓN, JUNIO 2014

“Dios hizo los números enteros, el resto es obra del hombre”

LEOPOLD KRONECKER

DEDICATORIA

Dedico con mucho amor este trabajo significativo y final ***a Dios*** por ser mi luz y mi guía en todos los aspectos de la vida.

A mi querida madre ***Margarita Frank***, que ha sido para mí, mi mejor amiga y mi modelo perfecto a seguir.

A mi querido padre ***Gustavo Obando***, por darme ese apoyo incondicional y regalarme un poquito de su amor.

A mis hermanas la ***Nerey*** y *la* ***Naimita****,* a mi único hermanito ***Josué***, y a mi querida abuelita Doña ***Concepción Escobar***, que me estuvieron apoyando y animando todo este tiempo.

Jorge A. P. Frank

AGRADECIMIENTO

Estoy muy agradecido grandemente con nuestro ***Padre Celestial*** por haberme dado esa oportunidad de culminar mi estudio superior.

A ***mis Padres*** que sin ellos no hubiera concluido con éxito lo que un día comencé. Que con sus manos siempre extendidas pude ser ese alguien anhelado.

A ***mi tutora Msc. Ángela Altamirano*** por brindarme su valioso tiempo, por ser una persona de mucho humanismo, y por enseñarme el camino de la sabiduría con mucha inteligencia y amor.

A ***mis queridos y prestigiosos maestros*** por hacer la gran labor de transformarme en la persona que hoy soy. Que por cierto sus consejos y regaños me han hecho más fuerte.

Me siento finalmente, muy agradecido con todos aquellos que estuvieron involucrados de alguna u otra manera con mi estudio superior. Que junto con ustedes supe vivir esa bonita experiencia de la vida.

Jorge A. P. Frank

INDICE

INTRODUCCION……..……1

OBJETIVOS……..……..3

CAPITULO 1. CONCEPTOS BASICOS

1.1 Introducción…….……..……4

1.2 Funciones diferenciables……..……..……..……4

1.3 Vectores y matrices…….……....…....7

1.4 Problema de valor inicial para Ecuaciones Diferenciales Ordinarias……..……...……..…15

1.5 Aspectos Generales de MATLAB…….……...……..…20

CAPITULO 2. MÉTODOS NUMÉRICOS

2.1 Introducción…….……....…..…22

2.2 Método de Newton Raphson…..…….…….....22

2.3 Método Directo de Eliminación Gaussiana con Sustitución hacia Atrás…….……..…....26

2.4 Método Iterativo de Gauss-Seidel……..…….....31

2.5 Método de Newton para sistemas de ecuaciones no lineales……..35

2.6 Método de Runge-Kutta de cuarto orden……..39

2.7 Método de Predictor-Corrector de Adams de cuarto orden……..….43

2.8 Método de Runge-Kutta para sistema de ecuaciones diferenciales de primer orden……..……..……..48

CAPITULO 3. APLICACIONES

3.1 Caso especial para Ecuaciones de una variable……..……..…….58

3.2 Caso especial para Sistema de Ecuaciones Lineales……..…….....60

3.3 Caso especial para Sistema de Ecuaciones no Lineales…….....62

3.4 Caso especial para Ecuaciones y Sistema de Ecuaciones Diferenciales Ordinarias……..……..64
CONCLUSIONES……..…….……..…..70
BIBLIOGRAFÍA…….…….……72
ANEXOS
Anexo Nº 1. Programa de Newton-Raphson…….……..…….……74
Anexo Nº 2. Programa de Eliminación Gaussiana con Sustitución Hacia Atrás……76
Anexo Nº 3 Programa de Gauss-Seidel……..……...78
Anexo Nº 4 Programa de Newton para Sistemas de Ecuaciones no Lineales……..…..……...……..80
Anexo Nº 5 Programa de Runge-Kutta de orden cuatro…...……..…..……..82
Anexo Nº 6 Programa de Predictor-Corrector de Adams de cuarto orden…...84
Anexo Nº 7 Programa de Runge-Kutta para Sistema de Ecuaciones Diferenciales de primer orden……..……..……..86
Anexo Nº 8 Programa de Runge-Kutta para Sistema de Ecuaciones Diferenciales de primer orden……..……..……..86
Anexo Nº 9. Tablas de ventajas y desventajas de los métodos numéricos estudiados para la solución de problemas modelados matemáticamente……..……92
Anexo Nº 10. Programa de Newton-Raphson para el caso 1…….……..…….95
Anexo Nº 11. Programa de Eliminación Gaussiana con Sustitución Hacia Atrás para el caso 2……..……97
Anexo Nº 12. Programa de Gauss-Seidel para el caso 2……....99
Anexo Nº 13. Programa de Newton para Sistemas de Ecuaciones no Lineales para el caso 3..……..…..……...……101

Anexo Nº 14. Programa de Runge-Kutta de orden cuatro para el caso 1…...103

Anexo Nº 15. Programa de Predictor-Corrector de Adams de cuarto orden

Para el caso 1……..……105

Anexo Nº 16 Programa de Runge-Kutta para Sistema de Ecuaciones

Diferenciales de primer orden para el caso 2……...……..107

INTRODUCCIÓN

En este mundo de constantes cambios, un problema matemático es un modelo que se deduce de un fenómeno físico. El modelo matemático consecuente puede representarse por medio de ecuaciones: de una variable, de múltiple variables; lineales, o no lineales, o puede ser un problema que se exprese en forma de ecuaciones diferenciales ordinarias, y que estén sujetos a condiciones de valores iniciales, todos ellos pueden ubicarse en una dimensión finita o infinita.

En los casos en que es prácticamente difícil encontrar la solución analítica o es verdaderamente costoso de resolver el problema matemático propuesto, optamos por obtener una aproximación a la solución mediante procedimientos numéricos, de manera eficiente y con suficiente precisión.

El área de la matemática en donde se realiza este tipo de cálculos numéricos se llama Análisis Numérico. Más formalmente, el **Análisis Numérico** diseña métodos numéricos que aproximan con precisión fijada las soluciones de problemas modelados matemáticamente que no se pueden resolver con exactitud.

El **modelo matemático** se implementa directamente en algoritmos numéricos en un computador, que permite estudiar las características del proceso uniendo las ventajas de la teoría y de la iteración numérica. Un **método numérico** es un algoritmo que permite dar en un número finito de pasos, una aproximación numérica, a la solución de un problema con un error prefijado que puede ser tan pequeño como se desee.

Un programa matemático computacional muy importante para implementar con mucha rapidez los algoritmos numéricos es el **MATLAB**, el cual, es un software de licencia ampliamente utilizado por académicos e investigadores matemáticos, con el fin de obtener soluciones aproximadas de problemas complicados de resolver analíticamente.

En este trabajo se intenta mostrar el marco teórico y las aplicaciones de siete métodos numéricos que comúnmente se mencionan en la actualidad; son métodos francamente eficientes. También se da a conocer las ventajas y desventajas de cada método.

El trabajo se presenta en tres capítulos. En el primer capítulo se introduce los conceptos y resultados básicos que facilitan la comprensión de los métodos numéricos analizados. En el segundo capítulo, se estudia la teoría de siete métodos numéricos muy utilizados así como sus alcances y limitaciones, y en el tercer capítulo se ilustran aplicaciones de estos métodos. Los programas correspondientes a los algoritmos de dichos métodos se encuentran en los anexos con la corrida de los ejemplos e implementados en MATLAB.

.

OBJETIVOS

OBJETIVO GENERAL

- Mostrar las aplicaciones de los métodos numéricos utilizando MATLAB para la implementación de algoritmos.

OBJETIVOS ESPECIFICOS

- Describir los métodos numéricos a utilizar de manera breve y concisa.
- Presentar algunos campos donde comúnmente se aplican estos métodos numéricos.
- Usar el software matemático MATLAB para elaborar los programas de los algoritmos de los métodos numéricos analizados.
- Ilustrar la corrida correspondiente en MATLAB de cada caso de aplicación considerado.

CAPITULO 1
CONCEPTOS BASICOS

1.1 INTRODUCCIÓN

El presente capitulo introduce los conceptos básicos que son de sumo valor para el entendimiento de los métodos numéricos del segundo capítulo. La mayoría de los métodos numéricos que abordaremos en el segundo capítulo consistirán en hallar la solución aproximada de problemas modelados matemáticamente, mediante procesos iterativos y a partir de aproximaciones iniciales bajo ciertos criterios de error.

Los conceptos introducidos son acerca de temas que se mencionan repetidamente en cualquier artículo de matemática avanzada, la idea parte desde las funciones diferenciables hasta una breve reseña descriptiva del software matemático MATLAB que utilizaremos para la ejecución los métodos numéricos del segundo capítulo.

Los teoremas se presentan sin sus demostraciones, ya que éstas pueden encontrarse en la mayoría de los textos avanzados de matemáticas, indicados en la bibliografía consultada.

1.2 FUNCIONES DIFERENCIABLES

Los conceptos elementales de límite y continuidad, juegan un papel muy significativo dentro del análisis numérico. Por lo que, a continuación citaremos dichos conceptos.

Definición 1.1 (Limite de una Función)

Sea f una función definida en un conjunto X de números reales. Se dice que, f tiene como **límite** al numero L en el punto x_0, denotado por $\lim_{x\to x_0} f(x) = L$, si dado cualquier número real $\varepsilon > 0$, existe un número real $\delta \neq 0$ tal que $|f(x) - L| < \varepsilon$, siempre que $x \in X$ y $0 < |x - x_o| < \delta$.

Definición 1.2 (Función Continua)

Sea f una función definida en un conjunto X de números reales y $x_0 \epsilon X$; se dice que f es **continua** en el punto x_0 si $\lim_{x \to x_0} f(x) = f(x_0)$. Se dice que f es continua en X si es continua en cada punto de X.

La idea de lo que es la convergencia de una sucesión está asociada directamente al concepto de límite, por eso se da la siguiente definición.

Definición 1.3 (Sucesión Convergente)

Sea $\{x_n\}_{n=1}^{\infty}$ una sucesión infinita de números reales o complejos. Se dice que

la **sucesión converge** al número x (llamado el límite) si, para toda $\varepsilon > 0$, existe un entero positivo $N(\varepsilon)$ tal que $n > N(\varepsilon)$ implica que $0 < |x_n - x| < \varepsilon$.

El siguiente teorema está estrechamente relacionado con los conceptos de convergencia y continuidad.

Teorema 1.1 (Convergencia y Continuidad)

Si f es una función definida en un conjunto X de números reales y $x_0 \epsilon X$, entonces las siguientes expresiones son equivalentes:

a) f es continua en el punto x_0;

b) si $\{x_n\}_{n=1}^{\infty}$ es cualquier sucesión en X convergente a x_0, entonces

$$\lim_{n \to \infty} f(x_n) = f(x_0).$$

Supongamos que las funciones con las que vamos a trabajar en el segundo capítulo, son continuas; ya que esto es un pequeño requisito para obtener un comportamiento predecible. Mientras que si las funciones son discontinuas pueden interrumpirse en los puntos de interés y eso causa dificultades en aproximar a la solución del problema propuesto. De tal manera que el teorema 1.1 es uno de los requisitos de trabajo para los métodos numéricos del segundo capítulo.

Definición 1.4 (Función Diferenciable)

Si f es una función definida en un intervalo abierto que contiene al punto x_0, se dice que f es **diferenciable** en x_0 si:

$$\lim_{x \to x_0} \frac{f(x)-f(x_0)}{x-x_0}$$

existe. Cuando este límite existe se denota por $f'(x_0)$ y se conoce como la derivada de f en x_0. Se dice que una función es diferenciable en X si la función tiene derivadas en cada punto del conjunto X.

Teorema 1.2 (Continuidad y Diferenciabilidad)

Si la función f es diferenciable en el punto x_0, entonces f es continua en x_0.

Para facilitar la terminología utilizada se establece la siguiente notación:

Notación

- La notación $\lim_{n \to \infty} x_n = x$, o $x_n \to x$ cuando $n \to \infty$, significa que la sucesión $\{x_n\}_{n=1}^{\infty}$ converge a x.
- La notación $C(X)$ denota el conjunto de todas las funciones reales continuas sobre X.
- El conjunto de todas las funciones continuas en el intervalo cerrado $[a, b]$ se denota por $C[a, b]$.
- El conjunto de todas las funciones que tienen n derivadas continuas en X se denota por $C^n(X)$.
- El conjunto de funciones que tiene derivadas de todos los órdenes en cada punto de X se expresa $C^{\infty}(X)$.
- El símbolo $f \in C[a, b]$ significa que la función $f : [a, b] \to R$ es continua en el intervalo $[a, b]$.
- Si $f \in C^n(X)$, la función f se dice que es de clase n en X y significa que todas las derivadas de f hasta orden n existen y son continuas en X.

Ahora, se presentan los teoremas fundamentales en la deducción de métodos para la estimación del error.

Teorema 1.3 (Teorema del Valor Medio)

Si $f \in C\,[a,b]$ y f es diferenciable en (a,b), entonces existe un numero c, y $a < c < b$, tal que

$$f'(c) = \frac{f(b) - f(a)}{b - a}$$

El siguiente teorema describe a los polinomios de Taylor. El primer método del segundo capítulo usa este teorema para su fundamento.

Teorema 1.4 (Teorema de Taylor)

Supongamos que $f \in C^n[a,b]$ y $f^{(n+1)}$ existe en $[a,b]$. Sea $x_0 \in [a,b]$. Para toda $x \in [a,b]$ existe $\xi(x)$ entre x_0 y x tal que:

$$f(x) = P_n(x) + R_n(n)$$

donde

$$P_n(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \ldots + \frac{f^n(x_0)}{n!}(x - x_0)^n$$
$$= \sum_{k=0}^{n} \frac{f^k(x_0)}{k!}(x - x_0)^k$$

y

$$R_n(x) = \frac{f^{n+1}(\xi(x))}{(n+1)!}(x - x_0)^{n+1}.$$

A $P_n(x)$ se le llama el **polinomio de Taylor** de grado n para f alrededor del punto x_0 y a $R_n(x)$ se le llama el **termino del residuo** (o el error de truncamiento) asociado a $P_n(x)$. Es llamada la serie de Taylor cuando se toma a $P_n(x)$ como la serie infinita de su límite cuando $n \to \infty$ y como el polinomio de Maclaurin cuando $x_0 = 0$. El término de **error de truncamiento** generalmente se refiere al error involucrado al usar sumas finitas o truncadas para aproximar la suma de una serie infinita.

1.3 VECTORES Y MATRICES

Definición 1.5 (Vector)

Un **vector** real x de dimensión n es un conjunto ordenado de n números reales que se escribe así: $x = (x_1, x_2, \dots, x_n)$.

A los números $x_1, x_2, \dots, x_n$ se les llaman componentes (o coordenadas) de x.

El conjunto formado por todos los vectores de dimensión n se le llama espacio de dimensión n y se denota por R^n.

En el momento en que un vector es utilizado para denotar un punto o una posición en el espacio, es llamado vector de posición**,** y cuando indica un movimiento entre dos puntos del espacio es llamado vector de desplazamiento.

Definición 1.6 (Igualdad de dos Vectores)

Sean $x = (x_1, x_2, \dots, x_n)$ y $y = (y_1, y_2, \dots, y_n)$ dos vectores. Se dice que los vectores x e y son iguales cuando sus respectivos componentes son iguales, es decir que,

$$x = y \quad \text{si y solo si} \quad x_j = y_j \quad \text{para } j = 1,2, \dots, n.$$

Definición 1.7 (Aritmética de Vectores)

Las operaciones elementales de vectores, se definen a continuación.

1. La suma de dos vectores x e y se calcula sumando un componente de x con un componente de y, o sea,
$$x + y = (x_1 + y_1, x_2 + y_2, \dots, x_n + y_n).$$

2. El opuesto del vector x es el vector que se obtiene reemplazando cada coordenada por su opuesta:
$$-x = (-x_1, -x_2, \dots, -x_n).$$

3. La diferencia de dos vectores x e y se obtiene restando un componente de x con un componente de y, o sea,
$$x - y = (x_1 - y_1, x_2 - y_2, \dots, x_n - y_n).$$

4. Dos vectores de dimensión n cumplen con la propiedad algebraica:
$$y - x = y + (-x).$$

5. Si c es un número real (un escalar), se define el producto de c y x como:
$$cx = (cx_1, cx_2, \dots, cx_n)$$

6. La combinación lineal de dos vectores x e y con dos escalares c y d es la suma ponderada $cx + dy$, que resulta así,
$$cx + dy = (cx_1 + dy_1, cx_2 + dy_2, \dots, cx_n + dy_n).$$

7. El producto escalar de dos vectores x e y es un escalar (un numero real) definido por la relación
$$x \cdot y = x_1y_1 + x_2y_2 + \cdots + x_ny_n$$

8. En el espacio de dimensión n el vector 0 se define así;
$$0 = (0, 0, \dots 0)$$

Si queremos definir una distancia en la dimensión n de los números reales, o bien, R^n aplicamos la siguiente definición.

Definición 1.8 (Norma Vectorial)

Una norma vectorial en R^n, es una función $\|\cdot\|$ de R^n en R con las siguientes propiedades:

i) $\|x\| \geq 0$ para toda $x \in R^n$,
ii) $\|x\| = 0$ si y solo si $x = (0,0, \dots,0)^t \equiv 0$,
iii) $\|\alpha x\| = |\alpha|\|x\|$para toda $\alpha \in R$ y $x \in R^n$,
iv) $\|x + y\| \leq \|x\| + \|y\|$para toda $x, y \in R^n$.

Para nuestro objetivo solo necesitaremos dos normas específicas en R^n. Las que son l_2 y l_∞ , que más adelante, aclararemos.

Al saber que los vectores en R^n son vectores columna, es conveniente emplear la notación de la transpuesta cuando se representa por medio de sus componentes. Un **vector columna** se representa así,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},$$

que es equivalente a la notación de la **transpuesta** que se escribe como $x = (x_1, \quad x_2, \ldots, \quad x_n)^t$.

Definición 1.9 (Norma Euclideana)

Las normas l_2 y l_∞ del vector $x = (x_1, \quad x_2, \ldots, \quad x_n)^t$ están definidos por

$$\|x\|_2 = \{\sum_{i=1}^{n} x_i^2\}^{1/2} \quad \text{y} \quad \|x\|_\infty = \max_{1 \le i \le n} |x_i|.$$

La norma l_2 se llama **norma Euclideana** del vector x, que representa la distancia respecto al origen.

Un ejemplo ilustrativo de la norma Euclideana es, la norma l_2 del vector $x = (x_1, x_2, x_3)^t$ que indica la longitud del segmento de la recta que une los puntos $(0, 0, 0)$ y $(x_1, \ x_2, \ x_3)$, en R^3.

Como la norma de un vector da una medida de la longitud entre un vector arbitrario y el vector cero del origen, la distancia entre estos dos vectores se puede definir mediante la diferencia de los dos vectores; en la siguiente definición se plantea esta situación.

Definición 1.10

Sean $x = (x_1, \quad x_2, \ldots, \quad x_n)^t$ e $y = (y_1, \quad y_2, \ldots, \quad y_n)^t$ dos vectores en R^n, las distancias l_2 y l_∞ entre estos dos vectores x e y se define como:

$$\|x - y\|_2 = \{\sum_{i=1}^{n} (x_i - y_i)^2\}^{1/2} \qquad \text{y} \qquad \|x - y\|_\infty = \max_{1 \le i \le n} |x_i - y_i|.$$

Definición 1.11

Se dice que una sucesión de vectores $\{x^{(k)}\}$ en R^nconverge a x con respecto a la norma $\|\cdot\|$ si dado cualquier $\varepsilon > 0$, existe un entero $N(\varepsilon)$ tal que:

$$\|x^{(k)} - x\| < \varepsilon \quad \text{para todo } k \ge N(\varepsilon)$$

Teorema 1.5

La sucesión de vectores $\{x^{(k)}\}$ en R^n converge a x con respecto a $\|\cdot\|_\infty$ si y solo si el $\lim_{k\to\infty} x_i^{(k)} = x_i$ para cada $i = 1,2,\dots,n$.

Definición 1.12 (Matriz)

Una **matriz** $(n \times m)$ es un arreglo de números reales de la forma rectangular, en filas y en columnas. Una matriz con n filas y m columnas se dice que es de orden $n \times m$. Vamos a representar a una matriz con A mayúscula, mientras los a_{ij} indican cada uno de los elementos que forman la matriz A, de manera que escribimos:

$$A = [a_{ij}]_{n\times m} \quad \text{para} \quad 1 \le i \le n, \ \ 1 \le j \le m,$$

Siendo a_{ij} el elemento que ocupa la posición (i,j), o sea, que está en la i-esima fila y la j-esima columna de la matriz, es decir,

$$A = [a_{ij}]_{n\times m} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix}$$

Las filas de una matriz A de orden $n \times m$ son vectores de dimensión m, o sea,

$$V_i = (a_{i1}, a_{i2}, \dots, a_{im}) \quad \text{para} \ \ i = 1, 2, \dots, n.$$

Definición 1.13 (Igualdad de dos Matrices)

Sean $A = \left[a_{ij}\right]_{n\times m}$ y $B = \left[b_{ij}\right]_{n\times m}$ dos matrices del mismo orden $n \times m$. Se dice que las dos matrices A y B son iguales cuando sus elementos correspondientes son iguales; es decir,

$$A = B \ \text{ si y solo si, } \ a_{ij} = b_{ij} \quad \text{para } 1 \le i \le n, \ \ 1 \le j \le m.$$

Definición 1.14 (Aritmética de Matrices)

Las operaciones elementales de matrices se definen así,

1. La suma de dos matrices A y B del mismo orden $n \times m$ se calcula elemento a elemento; o sea,

$$A + B = \left[a_{ij} + b_{ij}\right]_{n\times m} \quad \text{para } 1 \le i \le n,\ \ 1 \le j \le m.$$

2. La opuesta de una matriz A se obtiene sustituyendo cada elemento por su opuesto:

$$-A = \left[-a_{ij}\right]_{n\times m} \quad \text{para } 1 \le i \le n,\ \ 1 \le j \le m.$$

3. La diferencia $A - B$ se forma restando elemento a elemento:

$$A - B = \left[a_{ij} - b_{ij}\right]_{n\times m} \quad \text{para } 1 \le i \le n,\ \ 1 \le j \le m.$$

4. Si c es un numero real (un escalar), definimos el producto (múltiplo escalar) cA como:

$$cA = \left[ca_{ij}\right]_{n\times m} \quad \text{para } 1 \le i \le n,\ \ 1 \le j \le m.$$

5. Si p y q son escalares, la suma ponderada $pA + qB$ se le llama combinación lineal de las matrices A y B, y se tiene

$$pA + qB = \left[pa_{ij} + qb_{ij}\right]_{n\times m} \quad \text{para } 1 \le i \le n,\ \ 1 \le j \le m.$$

6. La matriz cero de orden $n \times m$ es aquella cuyos elementos son todos cero:

$$0 = [0]_{n\times m}.$$

Definición 1.15 (Multiplicación de dos Matrices)

Sean $A = [a_{ik}]_{n\times m}$ una matriz con n renglones y m columnas, $B = \left[b_{kj}\right]_{m\times p}$ otra matriz con m renglones y p columnas; el producto de AB se define como la matriz C de dimensión $n \times p$:

$$AB = C = \left[c_{ij}\right]_{n\times p} \text{ donde } c_{ij} = \left(\sum_{k=1}^{m} a_{ik}b_{kj}\right)_{n\times p}$$

Definición 1.16 (Matriz Identidad)

Sea $I = [\varphi_{ij}]_{n\times n}$ la **matriz identidad** de orden n, donde $\varphi_{ij} = \begin{cases} 1 \ \ si \ \ i = j \\ 0 \ \ si \ \ i \neq j \end{cases}$.

Observe que para toda matriz A de dimensión $n \times n$ se cumple lo siguiente: $AI = IA = A$.

Definición 1.17 (Transpuesta de una Matriz)

Sea A una matriz de dimensión $n \times m$, la transpuesta de A se representa A^t, es decir, que las filas de A se convierten en las columnas de su transpuesta. O sea,

$$A^t = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \vdots & \vdots & & \vdots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{bmatrix}$$

Definición 1.18 (Inversa Multiplicativa de una Matriz)

Sea A una matriz cuadrada $n \times n$ y sea B otra matriz cuadrada $n \times n$ tal que $AB = BA = I$ en donde I es la identidad multiplicativa. Se dice que la matriz B es la inversa multiplicativa de A denotada por $B = A^{-1}$.

Definición 1.19 (Matriz no Singular)

Sea A una matriz cuadrada $n \times n$. Si el $detA \neq 0$, entonces A es no singular. Pero, si el $detA = 0$, entonces A es singular.

El siguiente teorema describe una condición necesaria y suficiente para que una matriz tenga inversa multiplicativa.

Teorema 1.6

Se dice que la matriz cuadrada A tiene una inversa multiplicativa A^{-1} si y solo si A es no singular.

El teorema anterior detalla que la no singularidad implica que A tiene inversa.

Para determinar la distancia entre matrices del tipo $n \times n$ se requiere de inmediato el concepto de la norma de una matriz, que a continuación, definiremos.

Definición 1.20 (Norma Matricial)

Una norma matricial sobre el conjunto de todas las matrices de $n \times n$ es una función de valor real $\|\cdot\|$ definida en este conjunto y que satisface para todas las matrices A y B de $n \times n$ y todos los números reales α:

(i) $\|A\| \geq 0$.
(ii) $\|A\| = 0, \leftrightarrow A$ es 0, la matriz con todas las entradas ceros.
(iii) $\|\alpha \cdot A\| = |\alpha|\|A\|$.
(iv) $\|A + B\| \leq \|A\| + \|B\|$.
(v) $\|A \cdot B\| \leq \|A\|\|B\|$

La distancia entre las matrices A y B de $n \times n$ respecto a esta norma matricial es $\|A - B\|$.

Aunque las normas matriciales se pueden obtener de varias formas, pero únicamente consideremos las normas vectoriales l_2 y l_∞.

Teorema 1.7 (Norma Matricial Natural)

Si $\|\cdot\|$ es una norma vectorial de R^n, entonces

$$\|A\| = \max_{\|x\|=1}\|Ax\| \quad \text{es una norma matricial.}$$

A esta norma se le llama **norma matricial natural** asociada con la norma vectorial.

Para cualquier $z \neq 0$, tenemos que $x = z/\|z\|$ es un vector unitario. Por lo tanto,

$$\max_{\|x\|=1}\|Ax\| = \max_{\|x\|=1}\left\|A\left(\frac{z}{\|z\|}\right)\right\| = \max_{\|x\|=1}\frac{\|Az\|}{\|z\|},$$

Y podemos escribir en forma alternativa

$$\|A\| = \max_{z \neq 0} \frac{\|Az\|}{\|z\|}$$

El siguiente resultado es consecuencia de esta representación $\|A\|$.

Corolario 1.1

Para todo vector $z \neq 0$, matriz A y cualquier norma natural $\|\cdot\|$, tenemos

$$\|Az\| \leq \|A\| \cdot \|z\|$$

La extensión máxima es la norma de la matriz y se representa así:

$$\|A\|_\infty = \max_{\|x\|_\infty = 1} \|Ax\|_\infty, \quad \text{la norma } l_\infty$$

y

$$\|A\|_2 = \max_{\|x\|_2 = 1} \|Ax\|_2, \quad \text{la norma } l_2.$$

Teorema 1.8

Si $A = [a_{ij}]$ es una matriz cuadrada $n \times n$, entonces

$$\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^{n} |a_{ij}|.$$

1.4 PROBLEMA DE VALOR INICIAL PARA ECUACIONES DIFERENCIALES ORDINARIAS

En la mayor parte de los problemas matemáticos que se modelan como una ecuación diferencial ordinaria se trata de resolver un problema de valor inicial y que posteriormente, cuando tenga la resolución de la ecuación diferencial original a otra ecuación diferencial simplificada satisfaga la condición inicial dada.

Estas ecuaciones diferenciales pueden ser sencillas (de una ecuación diferencial para una función desconocida) o complejas (de un sistema de ecuaciones diferenciales ordinarias). Muchos problemas planteados con ecuaciones diferenciales son difíciles de resolver analíticamente, para los cuales una alternativa de solución son los métodos numéricos.

Los métodos para resolver una ecuación diferencial o sistema de ecuaciones diferenciales que estudiaremos en el segundo capítulo tienen la solución aproximada mediante una condición inicial.

En las secciones 2.6 y 2.7 del segundo capítulo estudiaremos como aproximar la solución $y(t)$ de un problema de valor inicial para una ecuación diferencial ordinaria, que es representado de la siguiente manera:

$$y'(t) = \frac{dy}{dt} = f(t, y)$$

con la condición de valor inicial $y(a) = \alpha$.

En la sección 2.8 de un problema de valor inicial para un sistema de ecuaciones diferenciales ordinarias como sigue:

$$y_1'(t) = \frac{dy_1}{dt} = f_1(t, y_1, y_2, \dots, y_n),$$

$$\begin{aligned} y_2'(t) &= \frac{dy_2}{dt} = f_2(t, y_1, y_2, \dots, y_n), \\ &\vdots \qquad\qquad\qquad \vdots \\ y_n'(t) &= \frac{dy_n}{dt} = f_n(t, y_1, y_2, \dots, y_n), \end{aligned}$$

en donde, $a \leq t \leq b$; con las siguientes condiciones iniciales dadas:

$$\begin{aligned} y_1(a) &= \alpha_1, \\ y_2(a) &= \alpha_2, \\ &\vdots \qquad \vdots \\ y_n(a) &= \alpha_n. \end{aligned}$$

Definición 1.21 (Condición Lipschitz)

Se dice que una función $f(t, y)$ satisface una **condición Lipschitz** en la variable y en un conjunto $D \subset R^2$ si existe una constante $L > 0$ con la propiedad de que:

$$|f(t, y_1) - (t, y_2)| \leq L|y_1 - y_2|,$$

siempre que (t, y_1), $(t, y_2) \in D$. A la constante L se le llama constante de Lipschitz para f.

Definición 1.22

Se dice que un conjunto $D \subset R^2$ es convexo si siempre que (t_1, y_1) y (t_2, y_2) pertenecen a D, el punto $((1-\alpha)t_1 + \alpha t_2, (1-\alpha)y_1 + \alpha y_2)$ tambien pertenecen a D para cada α en $[0, 1]$.

Teorema 1.9

Supongamos que $f(t,y)$ está definida en un conjunto convexo $D \subset R^2$. Si existe una constante $L > 0$ con

$$\left|\frac{\partial f}{\partial y}(t,y)\right| \leq L, \text{ para toda } (t,y)\, \epsilon\, D$$

entonces f satisface una condicion de Lipschitz en D en la variable y con la constante L de Lipschitz.

Teorema 1.10

Supongamos que $D = \{(t,y)|a \leq t \leq b, -\infty < y < \infty\}$ y que $f(t,y)$ es continua en D. Si f satisface una condición de Lipschitz en D en la variable y, entonces el problema de valor inicial

$$y'(t) = f(t,y), \qquad a \leq t \leq b, \qquad y(a) = \alpha$$

tiene una solución única $y(t)$ para $a \leq t \leq b$.

Definición 1.23

Se dice que el problema de valor inicial

$$\frac{dy}{dt} = f(t,y), \qquad a \leq t \leq b, \qquad y(a) = \alpha,$$

es un **problema bien planteado** si:

1. El problema tiene una solución única, $y(t)$;
2. Para cualquier $\varepsilon > 0$, existe una constante positiva $k(\varepsilon)$ con la propiedad de que siempre que $|\varepsilon_0| < \varepsilon$ y $\delta(t)$ es continua con $|\delta(t)| < \varepsilon$ en $[a,b]$, existe una solución única, $z(t)$; al problema,

   $$\frac{dz}{dt} = f(t,z) + \delta(t), \qquad a \leq t \leq b, \qquad z(a) = \alpha + \varepsilon_0,$$

   con

   $$|z(t) - y(t)| < k(\varepsilon)\varepsilon, \qquad \text{para toda } a \leq t \leq b.$$

Teorema 1.11

Suponga que $D = \{(t, y)|a \leq t \leq b, -\infty < y < \infty\}$. Si f es continua y satisface la condición de Lipschitz en la variable y en el conjunto D, entonces el problema de valor inicial

$$\frac{dy}{dt} = f(t, y), \qquad a \leq t \leq b, \qquad y(a) = \alpha,$$

es bien planteado.

1.5 ASPECTOS GENERALES DE MATLAB

Matrix laboratory en forma abreviada **MATLAB** es el software fundado por los analistas numéricos *Clere Moler* y *John N. Little*. Este programa se desarrolló básicamente como un laboratorio de matrices. Además, tiene su propio lenguaje de programación y es bastante amplio.

MATLAB tiene diversas funciones y operadores que accede apropiadamente la ejecución de los métodos numéricos. Otra de las cosas es que en MATLAB se puede realizar variedad de gráficos en dos y tres dimensiones.

Las principales características de MATLAB son los cálculos predefinidos con vectores y matrices como:

- Aritmética de vectores y matrices
- Inversión de matrices y análisis de valores y de vectores propios
- Aritmética compleja y operaciones con polinomios
- Cálculos estadísticos
- Despliegue de gráficas
- Diseño de sistemas de control
- Modelos de proceso de ajuste a partir del análisis de datos

MATLAB tiene diversas cajas de herramientas que proporcionan funciones especializadas. Estas incluyen procesamiento de señales, sistemas de control, identificación de sistemas, optimización y estadísticas.

A continuación, tendremos a disposición algunas características de cómo se puede aprender MATLAB.

1. Manipulación de MATLAB

La manipulación es sencilla haga dos veces clic en el icono MATLAB y espere que se abra una ventana, luego le aparecerá (») esto significa que está preparado para recibir instrucciones a ejecutar.

La parte en donde se escribe los programas numéricos es en *File*, entra en *File;* luego en *New* y por consiguiente en *M-File* y es alli donde puede escribir el programa de cualquier método que desee, que por ultimo lo puede correr en la ventana inicial llamado *MATLAB Comand Window.*

Para limpiar la ventana inicial escriba la función *clc*, de igual modo, para salir del programa solo escriba *quit* o elige *Exit MATLAB* en el menú *File* y para informarse de alguna función escriba el comando *Help.*

2. Asignación de valores a las variables

Como en otros lenguajes de computación es similar la asignación de valores a las variables. Por ejemplo: $h = 2$ o $Z = 4$, pero el problema es que en MATLAB asignar de esa manera implica repetición; por lo tanto se le pone punto y coma para que la repetición se elimine. Por ejemplo: $h = 2;$.

El nombre como b no es lo mismo B, tanto b y B son definitivamente distintos.

Un vector renglón se puede asignar así: $a = [\,1\ 2\ 3\ 4\ 5\,]$.

Un vector columna se introduce: $b = [1\ ;\ 2\ ;\ 3\ ;\ 4\ ;\ 5\,]$

o transponiendo el vector renglón con el operador (’): $b = [\,1\ 2\ 3\ 4\ 5\,]$’

Una matriz de valores en tres dimensiones se asigna como sigue:

$A = [\,1\ \ 2\ \ 3\ ;\ 4\ \ 5\ \ 6\ ;\ 7\ \ 8\ \ 9\,]$

Los valores numéricos que fueron almacenados por una variable pueden buscar tecleando el nombre de la variable. Por ejemplo: b o A.

3. Operaciones elementales de matemáticas

Para llevar a cabo las operaciones de cantidades escalares se hace directo.

Los operadores comunes son:

$+ \quad -$	Adición y sustracción
$* \quad /$	Multiplicación y división
^	Exponenciación
\	División por la izquierda (se aplica a matrices)

Estos operadores funcionan como en las calculadoras. Por ejemplo: $5 * pi$. También hace cálculos de variables escalares reales: $z = pi/2,\ z$^3.

Los cálculos de cantidades complejas operamos así. Por ejemplo: antes asignamos la variable $i = sqrt(-1)$ donde i es la raíz cuadrada de -1, ahora asignemos un número complejo, $x = 4 + i * 9$ y calculemos; $2 * x,\ \ 4/x,\ \ x$^2, $x + z,\ \ x - z$.

El enfoque principal de MATLAB está directamente sujeto a cálculos matriciales. El producto interno de dos vectores (producto punto) se calcula usando el operador $(.*)$. Por ejemplo: $a.* b$.

Para la multiplicación de un vector por una matriz, supongamos que, a es el vector y A es la matriz.
$a = [1\ \ 2\ \ \ 3\ ]$ y $\ A = [\ 1\ \ 2\ \ 3\ ;\ 4\ \ 5\ \ 6\ ;\ 7\ \ 8\ \ 9\ ]$

La multiplicación es $a * A$.

Para la multiplicación de una matriz por otra matriz se lleva a cabo de forma similar: $A * B$ con $B = [\ 3\ 4\ 5\ ;\ 1\ 2\ 9\ ;\ 5\ 4\ 9]$. Con escalares así: A/pi.

Otra de las cosas es que en MATLAB A^2 es una operación que se lleva a cabo consigo misma.

4. Funciones predefinidas

MATLAB tiene una amplia colección de funciones predefinidas. Una de las propiedades importantes es que operan directamente sobre cantidades vectoriales y matriciales. Por ejemplo: $log(A)$.

La mayoría de las funciones como $sqrt,\ abs,\ sin,\ acos,\ tanh,\ exp$ incluyendo $\log(A)$ operan en forma de arreglo. De igual forma, la exponencial y la raíz cuadrada tienen definiciones de matrices. Lo que hace MATLAB es evaluar la versión matricial cuando se agregue la letra m al nombre de la función. Por ejemplo $sqrtm(A)$.

5. Graficas en MATLAB

Hablar de graficas MATLAB tiene mucha similitud al de un programa en una hoja de cálculo. Pues, las graficas se crean rápidas y de manera conveniente. Por ejemplo: para crear una grafica de los arreglos t, y se introduce $plot(t, y)$.

Luego, para personificar su grafica se hace

$$title('grafica\ de\ y\ contra\ t')$$
$$xlabel('valores\ de\ t')$$
$$ylabel('valores\ de\ y')$$
$$grid$$

La grafica aparece en otra ventana y puede imprimirse o transferirse a través del portapapeles (PC con Windows o Mac) a otros programas.

Existen otras características de las graficas que serán de utilidad; trazo de graficas de objetos en lugar de líneas, familias de curvas, trazo de graficas en el plano complejo, ventanas de graficas múltiples, graficas log, log o semilog, graficas tridimensionales y graficas de contorno.

6. Resolución de polinomios

MATLAB operan sobre los arreglos como si las entradas fueran coeficientes o raíces de ecuaciones polinomiales. Por ejemplo: el polinomio $x^2 + x + 1 = 0$ como vector es $c = [\,1\ 1\ 1\,]$ y despues se almacenará en el arreglo r, es decir, $r = roots(c)$

Los coeficientes de un polinomio pueden calcularse a partir de las raíces con la función $poly, \quad poly(r)$

Un polinomio puede evaluarse para un valor dado de x. Por ejemplo, $polyval(c, 4.23)$.

Si otro polinomio, $x^3 + x^2 - 0.45x + 1$ se representa por el arreglo d,

$$d = [1\ 1\ -0.45\ 1]$$

Los dos polinomios pueden multiplicarse simbólicamente con la función convolución $(conv)$; para obtener los coeficientes del producto polinomial, se escribe: $cd = conv(c, d)$.

La función de $deconvolucion\ (deconv)$, sirve para dividir un polinomio entre otro; por ejemplo. $[q, r] = deconv(c, d)$.

El resultado q es el cociente y el resultado r es el residuo. Hay otras funciones que también son de mucha utilidad, como la función $residue$, que da la expansión en fracciones parciales.

7. Análisis estadístico

La caja de herramientas de estadística contiene muchas características para el análisis estadístico; sin embargo, los cálculos estadísticos comunes se realizan con el conjunto básico de funciones de MATLAB. Por ejemplo, $rand$ es una función con el que puede generar una serie de números aleatorios.

Por ejemplo; generar un vector v de longitud 21 con valores entre 0 y 100 a incrementos de 5; una distribución uniforme (0, 1) de longitud 20; un vector $unif$, uniforme cuya longitud sea la misma de v, y un grafico del mismo:

$v = 0{:}\,5{:}\,100;$
$rand(20)$
$unif = rand(size(v));$
$plot(unif)$

Se supone que estos son números distribuidos uniformemente con una media de 0.5, pues están entre 0 y 1; se comprueba al usar: $mean(unif)$; para la desviación típica con la función: $std(unif)$. Se puede hallar los valores mínimos y máximos mediante el uso de: $min(unif)$ y $max(unif)$. Por último para trazar el histograma de datos de una función, hacemos: $hist(unif, 10)$. donde 10 es el número de particiones.

Puede verse la amplia gama de usos del MATLAB, resumiendo que MATLAB opera interactivamente ejecutando una por una las instrucciones del que está operando, conforme va introduciendo la secuencia de declaraciones y datos. Otra de las cosas es que si usted guarda una serie de instrucciones como un guion (script) y puede correrlas como un programa interpretativo.

CAPITULO 2
MÉTODOS NUMÉRICOS

2.1 INTRODUCCIÓN

El presente capítulo tiene como objeto mostrar los métodos numéricos que vamos a estudiar, estos métodos se mencionan en muchos campos de la ciencia e ingeniera para hallar la solución aproximada de un problema matemático simple o complejo mediante el uso de procesos numéricos que aproximan las soluciones reales. Lo cierto es que estos métodos son eficaces, pero también poseen ciertas debilidades en algunas situaciones particulares.

Se presenta para cada situación uno o dos métodos de acuerdo a su mayor uso, y con el siguiente orden:

a) Para la solución de ecuaciones de una variable, el método de Newton-Raphson.
b) Para la solución de sistemas de ecuaciones lineales, el método de Eliminación Gaussiana con Sustitución hacia Atrás de manera directa y el método de Gauss-Seidel de manera iterativa.
c) Para el caso de sistemas de ecuaciones no lineales, el método de Newton.
d) Para la solución de ecuaciones diferenciales ordinarias con condiciones de valor inicial, el método de Runge-Kutta de cuarto orden y el método Predictor-Corrector de Adams de cuarto orden.
e) Para la solución de sistemas de ecuaciones diferenciales de primer orden el método Runge-Kutta para sistemas de ecuaciones diferenciales de primer orden.

2.2 METODO DE NEWTON-RAPHSON

Comenzamos el estudio de los métodos de este capítulo, con el primer método, que para muchos, es conocido. Se trata del **método de Newton-Raphson** (o simplemente *Newton*) en honor a los matemáticos *Isaac Newton* y *Joseph Raphson*.

El *método de Newton-Raphson* sin duda es uno de los mejores métodos numéricos iterativos que existe con la especialidad de hallar con precisión los ceros de una ecuación de la forma $f(x) = 0$. Hay diversas maneras de

establecer este método. La manera común de introducir está basada en el uso del reconocido *polinomio de Taylor*.

Supóngase que la función f es continuamente diferenciable dos veces en el intervalo $[a, b]$, es decir, $f \in C^2\,[a, b]$. Sea $x_0 \in [a, b]$ una aproximación de p tal que $f'(x_0) \neq 0$ y que $|p - x_0|$ es relativamente pequeño. Ahora, considere el polinomio de Taylor de primer orden para $f(x)$ alrededor de x_0.

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2}{2} f''\big(\xi(x)\big) \qquad (1)$$

donde $\xi(x)$ se encuentra entre x y x_0, y haciendo a $x = p$ y sustituyendo $f(p) = 0$, la ecuación (1) resulta,

$$0 = f(x_0) + (p - x_0)f'(x_0) + \frac{(p - x_0)^2}{2} f''\big(\xi(p)\big) \qquad (2)$$

Truncando la ecuación (2) despues del término de la primera derivada, o sea, despreciando a $(p - x_0)^{2,}$ se obtiene la siguiente versión aproximada:

$$0 = f(x_0) + (p - x_0)f'(x_0) \qquad (3)$$

Al despejar el valor de p de la ecuación (3) obtenemos,

$$p \approx x_0 - \frac{f(x_0)}{f'(x_0)} \qquad (4)$$

lo cual significa que p debe ser una mejor aproximación que x_0.

Al resultado de la ecuación (4) se le conoce con el nombre de **método de Newton-Raphson.** Para una representación general de este método se hace el uso de sucesiones, así la sucesión $\{p_n\}$ representa las n aproximaciones a la raíz:

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})} \quad n \geq 1 \qquad (5)$$

Aunque, francamente en muchas situaciones la formula (5) es muy eficiente, hay situaciones en las que se comporta de manera deficiente. Por ejemplo, en una situación particular puede que la convergencia sea demasiadamente lenta; o en el caso en que $f'(p_{n-1}) = 0$ para alguna n no podrá continuar la aplicación del método.

La convergencia del método de Newton-Raphson depende de la naturaleza de la función y de un buen valor inicial suficientemente cercano a la raíz como un

punto de arranque hacia la solución. A este método también se le conoce como un método abierto.

La fórmula (5) no sólo es para hallar las raíces reales de la ecuación $f(x) = 0$ en un campo real, sino también se pueden localizar las raíces complejas en un plano geométrico complejo $z = a + bi$, aunque es más complicada su búsqueda. Como suelen derivar las funciones de variables reales es posible también realizar la derivada sobre las funciones de variables complejas. El método se puede expandir en el campo complejo, con una aproximación inicial z_0 y generar una sucesión de puntos z_k que converjan a la raíz compleja.

ALGORITMO DEL METODO DE NEWTON-RAPHSON

Para encontrar la solución de la ecuación $f(x) = 0$ que está dada con una aproximación inicial p_0:

ENTRADA: La aproximación inicial p_0; La tolerancia TOL; El número máximo de iteraciones N_0.

SALIDA: La solución aproximada p o el mensaje de fracaso.

Paso 1 Tomar $i = 1$.

Paso 2 Mientras que $i \leq N_0$ seguir Paso 3-6.

Paso 3 Tomar $p = p_0 - f(p_0)/f'(p_0)$. (Calcular p_i).

Paso 4 Si $|p - p0| < TOL$ entonces
SALIDA (p); (Procedimiento completado satisfactoriamente)
PARAR.

Paso 5 Tomar $i = i + 1$.

Paso 6 Tomar $p_0 = p$ (Redefinir p_0).

Paso 7 SALIDA (El método fracaso después de N_0 iteraciones, $N_0 = ', N_0$);
(Procedimiento completado sin éxito)
PARAR.

EJEMPLO 1

Considere la ecuación no lineal:

$$e^x - 3x^2 = 0 \qquad (6)$$

con una tolerancia de 10^{-3} y con una aproximación inicial de $p_0 = 0.5$.

Para hallar la solución aproximada de esta ecuación aplique el método de Newton-Raphson.

Solución:

Dada la ecuación $f(x) = 0$ se tiene que $f(x) = e^x - 3x^2$. La primera derivada de la función es $f'(x) = e^x - 6x$.

Tanto el valor de la derivada y de la función original en cada punto p_{n-1} se sustituye en la ecuación del método para obtener el valor de p_n. Asi,

Empezando con el valor inicial $p_0 = 0.5$, le aplicamos la ecuación (5) para calcular p_1 y p_2 resultan así,

$$p_1 = p_0 - \frac{f(p_0)}{f'(p_0)} = 0.5 - \frac{f(0.5)}{f'(0.5)} = 0.5 - \frac{e^{0.5} - 3(0.5)^2}{e^{0.5} - 6(0.5)} = 1.165089$$

$$p_2 = p_1 - \frac{f(p_1)}{f'(p_1)} = 1.165089 - \frac{f(1.165089)}{f'(1.165089)}$$

$$= 1.165089 - \frac{e^{1.165089} - 3(1.165089)^2}{e^{1.165089} - 6(1.165089)} = 0.936227$$

Repitiendo el mismo proceso para calcular p_3y p_4 se obtienen que $p_3 = 0.910397$ y $p_4 = 0.910008$. En la tabla 1 se resumen los resultados con el error respectivo a este ejemplo aplicando el software de MATLAB que es una herramienta computacional básica para precisar los resultados aproximados.

i	p_i	$\lvert p_i - p_{i-1}\rvert$
1	1.165089	0.665089
2	0.936227	0.228863
3	0.910397	0.025830
4	0.910008	0.000389

Tabla 1

Con la tolerancia pedida de 10^{-3} se tiene que la solución aproximada de la ecuación (6) es 0.910008. Así, nos dimos cuenta que el método Newton-Raphson converge rápidamente a la raíz real.

En el anexo 1 se presenta el programa de Newton-Raphson implementado en MATLAB con la salida que corresponde a este ejemplo.

2.3 METODO DE ELIMINACION GAUSSIANA CON SUSTITUCION HACIA ATRÁS

A menudo se ve, un sistema lineal en problemas matemáticos, que puede ser resuelto por métodos sencillos y directos. A diferencia de los otros métodos, el método de esta sección tiene estas características.

Para resolver un sistema de ecuaciones lineales de la forma:

$$\begin{array}{l} E_1\colon\ a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ E_2\colon\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ E_k\colon\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_{n.} \end{array} \qquad (1)$$

donde las $x_1, x_2, \ldots, x_n$ son las n incógnitas, los a_{ij} con fila i por columna j son los coeficientes y los b_i como los valores numéricos del vector columna b.

Otra representación del sistema lineal (1) simplificada es $Ax = b$, donde A es la matriz de los coeficientes a_{ij}, la x como las incógnitas $x_1, x_2, \ldots, x_n$ y b como los b_i.

Hay varias maneras de resolver el sistema (1), pero nosotros haremos el uso de tres operaciones básicas; las cuales son,

1) La ecuación E_i puede multiplicarse por cualquier constante α distinto de cero, o sea, $\alpha \neq 0$ y la ecuación que resulte se emplea en vez de E_i. Esta acción se denota simbólicamente por $(\alpha E_i) \rightarrow (E_i)$.
2) La ecuación E_i puede multiplicarse por cualquier constante β , o sea, $\beta * E_i$ y sumarse a la ecuación E_j, o sea, $[(\beta * E_i) + E_j]$ la ecuación que resulte se emplea en vez de E_i. Esta acción se denota simbólicamente por $[(\beta * E_i) + E_j] \rightarrow (E_i)$.
3) El orden de las ecuaciones E_i y E_j pueden intercambiarse. Esta acción se denota simbólicamente por $(E_i) \leftrightarrow (E_j)$.

Una vez, aplicado las tres operaciones anteriores al sistema (1) se transforma a otro sistema de la forma triangular como el que sigue,

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1, \\ a'_{22}x_2 + \cdots + a'_{2n}x_n &= b'_2, \\ \ddots \quad \vdots \quad & \vdots \\ \ddots \quad \vdots \quad & \vdots \\ a'_{nn}x_n &= b'_n \end{aligned} \qquad (3)$$

donde, los a'_{ij} son los nuevos valores de los coeficientes de a_{ij} y los b'_i de los b_i. Por lo que, se despeja las incógnitas; empezando con la ultima ecuación luego sustituye a la de arriba, y así sigue el mismo procedimiento con las otras ecuaciones. A este procedimiento se le conoce como el *proceso de sustitución hacia atrás* (3).

Ahora, tendremos a disposición el método de esta sección, tiene un procedimiento más general el cual se llama el **método de eliminación gaussiana con sustitución hacia atrás.** La aplicación de este método consiste en realizar el proceso de (3) pero, haciendo el uso de matrices, ya que las matrices almacenan toda la información del sistema lineal para determinar la solución.

Construimos la matriz del sistema (1) simbolizando con A en donde se toman únicamente los coeficientes a_{ij} y con b tomando únicamente los valores del vector columna de los b_i.

$$A = \begin{bmatrix} a_{11} & a_{12} \cdots & a_{1n} \\ a_{21} & a_{22} \cdots & a_{2n} \\ \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} \cdots & a_{nn} \end{bmatrix} \qquad (4)$$

y $$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \qquad (5)$$

Luego, combinando (4) con (5), formamos la matriz aumentada llamándolo $\bar{A}$:

$$\bar{A} = [A \vdots b] = \begin{bmatrix} a_{11} & a_{12} \cdots & a_{1n} & \vdots & b_1 \\ a_{21} & a_{22} \cdots & a_{2n} & \vdots & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} \cdots & a_{nn} & \vdots & b_n \end{bmatrix} \qquad (6)$$

En (6) Hacemos un pequeño cambio para los valores de b, tomamos los elementos en la $(n+1)$-ésima columna como los valores de $b_i = a_{i,n+1}$ para toda $i = 1,2, \ldots n.$ y esto resulta,

$$\bar{A} = [A \vdots b] = \begin{bmatrix} a_{11} & a_{12} \cdots & a_{1n} & \vdots & a_{1,n+1} \\ a_{21} & a_{22} \cdots & a_{2n} & \vdots & a_{2,n+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} \cdots & a_{nn} & \vdots & a_{i,n+1} \end{bmatrix} \quad (7)$$

En la matriz (7) al aplicar el procedimiento general de resolución sucede que; siempre que el primer elemento $a_{11} \neq 0$, las operaciones correspondientes al paso $(E_j - (a_{j1}/a_{11})E_j) \rightarrow (E_j)$ se efectúan por cada $j = 2, 3, \ldots, n$ para eliminar el coeficiente de x_1 en cada uno de estos renglones.

Aunque se espera que los elementos de los renglones $2, 3, \ldots, n$ cambien para facilitar la notación denotamos nuevamente con a_{ij} el elemento del i-ésimo renglón y la j-ésima columna. Teniendo presente este concepto, aplicamos un procedimiento secuencial cuando $i = 2, 3, \ldots, n-1$ y realizamos la operación $(E_j - (a_{ji}/a_{ii})E_1) \rightarrow (E_j)$ para toda $j = i+1, i+2, \ldots, n$, a condición de que $a_{ii} \neq 0$.

Con ello se transforma en cero el coeficiente de x_i en cada renglón debajo de i-ésimo para todo los valores de $i = 1,2, \ldots, n-1$.

Despues de haber aplicado el procedimiento general a la matriz (7) esta toma la estructura de;

$$\bar{\bar{A}} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & \vdots & a_{1,n+1} \\ 0 & a_{22} & \cdots & a_{2n} & \vdots & a_{1,n+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn} & \vdots & a_{n,n+1} \end{bmatrix} \quad (8)$$

Formándose así, el nuevo sistema lineal triangular:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= a_{1,n+1} \\ a_{22}x_2 + \cdots + a_{2n}x_n &= a_{2,n+1} \\ \ddots \quad \vdots \quad & \vdots \\ \ddots \quad \vdots \quad & \vdots \\ + a_{nn}x_n &= a_{n,n+1}, \end{aligned} \quad (9)$$

En el sistema (9), el primer paso, es hallar la n-ésima ecuación para x_n, y obtenemos;

$$x_n = \frac{a_{n,n+1}}{a_{nn}} \quad (10)$$

El segundo paso, es resolver la $(n-1)$-ésima ecuación para x_{n-1} y al utilizar la incógnita x_n obtenemos,

$$x_{n-1} = \frac{a_{n-1,n+1} - a_{n-1,n}x_n}{a_{n-1,n-1}} \quad (11)$$

Y, así se puede continuar este proceso, obteniendo;

$x_i = \frac{a_{i,n+1} - a_{i,n}x_n - a_{i,n-1}x_{n-1} - \cdots - a_{i,i+1}x_{i+1}}{a_{ii}} = \frac{a_{i,n+1} - \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}}$, para cada $i = n-1, n-2, \ldots, 2, 1.$

(12)

Tomando las ecuaciones $(10), (11)$ y (12) el procedimiento que se le aplica es la *sustitución hacia atrás* para hallar los valores numéricos de las incógnitas $x_1, x_2 \ldots, x_n$.

ALGORITMO DEL MÉTODO DE ELIMINACIÓN GAUSSIANA CON SUSTITUCIÓN HACIA ATRÁS

Para resolver el sistema lineal de la forma $n \times n$

$$\begin{aligned} E_1&: \ a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = a_{1,n+1} \\ E_2&: \ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = a_{2,n+1} \\ &\vdots \qquad \vdots \qquad \vdots \qquad\qquad \vdots \qquad \vdots \\ E_n&: \ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = a_{n,n+1} \end{aligned}$$

ENTRADA El numero de incógnitas y ecuaciones n, la matriz aumentada $A = (a_{ij})$ donde $1 \leq i \leq n$ y $1 \leq j \leq n+1$.

SALIDA La solución $x_1, x_2, \ldots x_n$ o mensaje de que el sistema lineal no tiene solución única.

Paso 1 Para $i = 1, \ldots, n-1$ haga pasos 2-4 (proceso de eliminación)

Paso 2 Sea p el entero más pequeño con $i \leq p \leq n$ y $a_{pi} \neq 0$.
Si no puede encontrarse un entero p

entonces SALIDA ('no existe solución única');
PARAR.

Paso 3 Si $p \neq i$ entonces realice $(E_p) \leftrightarrow (E_i)$.

Paso 4 Para $j = i+1, \ldots, n$ haga pasos 5 y 6.

Paso 5 Tome $m_{ji} = a_{ji}/a_{ii}$.

Paso 6 Realice $(E_j - m_{ji}E_i) \rightarrow (E_j)$;

Paso 7 Si $a_{nn} = 0$ entonces SALIDA ('no existe solución única')

PARAR.

Paso 8 Tome $x_n = a_{n,n+1}/a_{nn}$. (Comience la sustitución hacia atrás.)

Paso 9 Para $i = n-1, \ldots, 1$ tome $x_i = \left[a_{i,n+1} - \sum_{j=i+1}^{n} a_{ij}x_j\right]/a_{ii}$.

Paso 10 SALIDA $(x_1, \ldots, x_n)$; (Procedimiento terminado exitosamente.)
PARAR.

EJEMPLO 2

Considere el siguiente sistema de cuatros ecuaciones con cuatro incógnitas:

$$\begin{aligned} E_1&: x_1 + 2x_2 - x_3 + 4x_4 = 1, \\ E_2&: \quad 4x_2 - 2x_3 + 0.25x_4 = 4, \\ E_3&: x_1 + 2x_2 + 6x_3 + 3x_4 = 3, \\ E_3&: 0.5x_1 + x_2 - 0.5x_3 + x_4 = 2. \end{aligned} \qquad (13)$$

Mediante el método de esta sección resuelva el sistema lineal de cuatro incógnitas con el fin de hallar las soluciones del sistema lineal.

Solución:

El sistema (13) lo transformamos en una matriz llamándolo A, tomando únicamente los coeficientes a_{ij} de las incógnitas y los b_i como b y los ubicamos sus respectivos lugares.

$$A = \begin{bmatrix} 1 & 2 & -1 & 4 \\ 0 & 4 & -2 & 0.25 \\ 1 & 2 & 6 & 3 \\ 0.5 & 1 & -0.5 & 1 \end{bmatrix} \quad \text{y} \quad B = \begin{bmatrix} 1 \\ 4 \\ 3 \\ 2 \end{bmatrix} \qquad (14)$$

Primer paso, se halla la matriz aumentada,

$$\bar{A} = \left[\begin{array}{cccc:c} 1 & 2 & -1 & 4 & 1 \\ 0 & 4 & -2 & 0.25 & 4 \\ 1 & 2 & 6 & 3 & 3 \\ 0.5 & 1 & -0.5 & 1 & 2 \end{array}\right] \qquad (15)$$

Aplicando las tres operaciones elementales con filas, efectuamos $(E_3 - E_1) \rightarrow (E_3)$, $(E_4 - 0.5\,E_1) \rightarrow (E_4)$ sobre la matriz (15) obtenemos,

$$\bar{A}^{(1)} = \left[\begin{array}{cccc:c} 1 & 2 & -1 & 4 & 1 \\ 0 & 4 & -2 & 0.25 & 4 \\ 0 & 0 & 7 & -1 & 2 \\ 0 & 0 & 0 & -1 & 1.5 \end{array}\right] \qquad (16)$$

Y por último, le aplicamos la sustitución hacia atrás para hallar los valores numéricos de las variables x_1, x_2, x_3 y x_4. Así tenemos,

$x_4 = -1.5,$

$$x_3 = \frac{2 + x_4}{7} = \frac{2 - 1.5}{7} = 0.071428571,$$

$$x_2 = \frac{4 - 0.25x_4 + 2x_3}{4} = \frac{4 - 0.25(-1.5) + 2(0.071428571)}{4} = 1.129464286\ ,$$

$$x_1 = 1 - 4x_4 + x_3 - 2x_2 = 1 - 4(-1.5) + 0.071428571 - 2(1.129464286$$
$$= 4.812499999.$$

∴ El sistema de ecuaciones lineales (13) tiene solución única para cada variable:

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} \rightarrow \begin{Bmatrix} 4.812499999 \\ 1.129464286 \\ 0.071428571 \\ -1.5 \end{Bmatrix}$$

Usando el instrumento computacional de MATLAB para hallar la solución del sistema, tenemos que:

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} \rightarrow \begin{Bmatrix} 4.81250 \\ 1.129464286 \\ 0.0714290 \\ -1.50 \end{Bmatrix}$$

En el anexo 2 se presenta el programa del método de Eliminación Gaussiana con Sustitución hacia Atrás implementado en MATLAB con la salida que corresponde a este ejemplo.

2.4 MÉTODO ITERATIVO DE GAUSS-SEIDEL

En la sección anterior estuvimos estudiando cómo se resuelve un sistema lineal de la forma $Ax = b$ con la técnica directa aplicando el método de Eliminación Gaussiana con Sustitución hacia Atrás, ahora aplicaremos un método que lo resuelve de manera iterativa el cual se llama **método iterativo de Gauss-Seidel** en honor a los matemáticos *Carl Friedrich Gauss* y *Philips Ludwig Von Seidel*.

Para resolver el sistema lineal de la forma $Ax = b$, mediante una técnica iterativa se comienza con un vector de aproximación inicial $x^{(0)}$ donde se repite un mismo proceso de mejora sobre la solución hasta que el resultado más reciente satisfaga ciertos requisitos y que el error sea relativamente pequeño.

El proceso genera una sucesión de vectores $\{x^{(k)}\}_{k=0}^{\infty}$ que converge al vector solución x del sistema.

Este método trae consigo un proceso que convierte al sistema lineal $Ax = b$ en otro análogo de la forma $x = Dx + c$ donde D es una matriz fija y c un vector . Una vez seleccionado el vector inicial $x^{(0)}$ la sucesión de vectores que converge a la solución aproximada se genera calculando:

$$x^{(k)} = Dx^{(k-1)} + c\ , \quad \text{para cada } k = 1,2,3,\ldots \qquad (1)$$

El uso de esta técnica iterativa para resolver un sistema lineal de dimensión pequeño, a veces, no es conveniente; en este caso, es más eficiente la aplicación de una técnica directa como el de la sección 2.3; ya que para sistemas grandes una técnica iterativa es más eficiente en la obtención de los resultados, ya que da una buena convergencia a la solución.

El método Gauss-Seidel es potente en el sentido de obtener los resultados muy cercanos a la solución, porque incorpora en sus cálculos valores más recientes. La sucesión de vectores que converge a la solución aproximada mediante el método de Gauss-Seidel se obtiene a partir de la ecuación siguiente:

$$x_i^{(k)} = \frac{-\sum_{j=1}^{i-1}\left(a_{ij}x_j^{(k)}\right)-\sum_{j=i+1}^{n}\left(a_{ij}x_j^{(k-1)}\right)+b_i}{a_{ii}}\text{, para } i = 1,2,\ldots,n \qquad (2)$$

La idea es que para calcular $x_i^{(k)}$ se hace uso de valores ya calculados recientemente, o sea, se incorpora $x_j^{(k)}$ para $1 \le j \le i-1$ y se usa $x_j^{(k-1)}$ para $i+1 \le j \le n$.

ALGORITMO DEL MÉTODO ITERATIVO DE GAUSS-SEIDEL

Para resolver el sistema lineal $Ax = b$ dada una aproximación inicial $x^{(0)}$:

ENTRADA: n es el número de ecuaciones e incógnitas; a_{ij} los elementos o coeficientes de la matriz, A la matriz de la condición $1 \le i, j \le n$; los números b_i, b con la condición $1 \le i \le n$; los elementos XO_i, $XO = x^{(0)}$ con la condición $1 \le i \le n$, TOL como la tolerancia; y N como el número máximo de iteraciones.

SALIDA: La solución aproximada $x_1, \ldots, x_n$ o el mensaje de que se excedió el numero de iteraciones.

Paso 1 Tomar $k = 1$.

Paso 2 Mientras $(k \le N)$ hacer los pasos de 3-6.

Paso 3 Para $i = 1, \dots, n$

Tome $x_i = \frac{\sum_{j=1}^{i-1} a_j x_j - \sum_{j=i+1}^{n} a_{ij} XO_j + b_i}{a_{ii}}$.

Paso 4 Si $\|x - XO\| < TOL$ entonces SALIDA $(x_1, \dots, x_n)$
(Procedimiento terminado exitosamente.)
PARAR.

Paso 5 Tome $k = k + 1$.

Paso 6 Para $i = 1, \dots, n$ tome $XO_i = x_i$.

Paso 7 SALIDA ('Número máximo de iteraciones excedido');
(Procedimiento terminado sin éxito.)
PARAR.

EJEMPLO 3

Considere el sistema de cuatro ecuaciones lineales con cuatro incógnitas:

$$\begin{aligned} 4x_1 + x_2 - x_3 + x_4 &= -2, \\ x_1 + 4x_2 - x_3 - x_4 &= -1, \\ -x_1 - 2x_2 + 5x_3 + x_4 &= 0, \\ x_1 - x_2 + x_3 + 3x_4 &= 1. \end{aligned} \qquad (3)$$

con un vector de aproximaciones iniciales dada de $\left\{x_1^{(0)}, x_2^{(0)}, x_3^{(0)}, x_4^{(0)}\right\} = \{0, 0, 0, 0\}^t$, aplique el método iterativo de Gauss-Seidel para resolver el sistema lineal.

Solución:

Primero, lo convierte a (3) a la forma $x^{(k)} = Dx^{(k-1)} + c$, es decir, se despejan las variables x_1, x_2, x_3 y x_4, a fin de obtener la solución aproximada de dicho sistema, y nos resulta:

$$x_1^{(k)} = -\frac{1}{4}x_2^{(k-1)} + \frac{1}{4}x_3^{(k-1)} - \frac{1}{4}x_4^{(k-1)} - \frac{1}{2},$$

$$x_2^{(k)} = -\frac{1}{4}x_1^{(k)} + \frac{1}{4}x_3^{(k-1)} + \frac{1}{4}x_4^{(k-1)} - \frac{1}{4},$$

$$x_3^{(k)} = \frac{1}{5}x_1^{(k)} + \frac{2}{5}x_2^{(k)} - \frac{1}{5}x_4^{(k-1)} + 0,$$

$$x_4^{(k)} = -\frac{1}{3}x_1^{(k)} + \frac{1}{3}x_2^{(k)} - \frac{1}{3}x_3^{(k)} + \frac{1}{3}.$$

Una vez que tengamos las variables despejadas, y estén listas para ser calculadas, comenzamos así:

Se calculan los componentes del vector $x^{(1)}$, obteniendo;

$$x_1^{(1)} = -\frac{1}{4}x_2^{(0)} + \frac{1}{4}x_3^{(0)} - \frac{1}{4}x_4^{(0)} - \frac{1}{2} = -0.5$$

$$x_2^{(1)} = -\frac{1}{4}x_1^{(1)} + \frac{1}{4}x_3^{(0)} + \frac{1}{4}x_4^{(0)} - \frac{1}{4} = -\frac{1}{4}(-0.5) + 0 + 0 - \frac{1}{4} = -0.125$$

$$x_3^{(1)} = \frac{1}{5}x_1^{(1)} + \frac{2}{5}x_2^{(1)} - \frac{1}{5}x_4^{(0)} + 0 = \frac{1}{5}(-0.5) + \frac{2}{5}(-0.125) - 0 + 0 = -0.150$$

$$x_4^{(1)} = -\frac{1}{3}x_1^{(1)} + \frac{1}{3}x_2^{(1)} - \frac{1}{3}x_3^{(1)} + \frac{1}{3} = -\frac{1}{3}(-0.5) + \frac{1}{3}(-0.125) - \frac{1}{3}(-0.125) + \frac{1}{3}$$
$$= 0.4999996$$

Y así sucesivamente, se va calculando las aproximaciones de los componentes de los vectores $x^{(2)}, x^{(3)}$ y $x^{(4)}$. En la tabla 2 se muestra el resultado usando el software de MATLAB con 8 iteraciones y con una tolerancia de 10^{-3} realizadas.

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$	$\left\|x^{(k)} - x^{(k-1)}\right\|$
1	-0.500000	-0.125000	-0.150000	0.508333	0.508333
2	-0.633333	-0.002083	-0.229167	0.620139	0.133333
3	-0.711806	0.025694	-0.256111	0.664537	0.078472
4	-0.736586	0.036253	-0.265723	0.679521	0.024780
5	-0.745374	0.039793	-0.269062	0.684743	0.008789
6	-0.748399	0.041020	-0.270220	0.686547	0.003025
7	-0.749447	0.041443	-0.270621	0.687170	0.001047
8	-0.749809	0.041589	-0.270760	0.687386	0.000362

Tabla 2

∴ Los valores aproximados de las variables del sistema lineal son:

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} \rightarrow \begin{Bmatrix} -0.749809 \\ 0.041589 \\ -0.270760 \\ 0.687386 \end{Bmatrix}$$

En el anexo 3 se presenta el programa de Gauss-Seidel implementado en MATLAB con la salida que corresponde a este ejemplo.

2.5 MÉTODO DE NEWTON

En la sección 2.2 estuvimos estudiando el *método de Newton-Raphson* para resolver problemas matemáticos que son modelados con una ecuación del tipo no lineal de la forma $f(x) = 0$. En esta sección, presentamos el **método de Newton para resolver sistemas de ecuaciones no lineales** con un procedimiento numérico que efectúa la transformación en una situación más compleja.

Para determinar las soluciones aproximadas de un sistema de ecuaciones no lineales de la forma:

$$\begin{aligned} f_1(x_1, x_2, \dots, x_n) &= 0 \\ f_2(x_1, x_2, \dots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, \dots, x_n) &= 0 \end{aligned} \qquad (1)$$

donde $f_1, f_2, \dots, f_n$ son funciones reales y diferenciables, las $x_1, x_2, \dots, x_n$ son las variables.

Estas funciones pueden definirse con una función F, que va de R^n en R^n como;

$$F(x_1, x_2, \dots, x_n) = [\, f_1(x_1, x_2, \dots, x_n),\ f_2(x_1, x_2, \dots, x_n),\ \dots,\ f_n(x_1, x_2, \dots, x_n)\,]^{\mathrm{t}}$$

y las x_i como el vector

$$x = (x_1, x_2, \dots, x_n)^t.$$

Las funciones $f_1, f_2, \dots, f_n$ son coordenadas de F y con esto, se puede interpretar el sistema de ecuaciones no lineales como un campo vectorial F, o sea, $F(x) = 0$, es decir;

$$F(x) = 0 \Leftrightarrow \begin{Bmatrix} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_1(x_1, x_2, \dots, x_n) = 0 \\ \vdots \\ f_{\mathrm{n}}(x_1, x_2, \dots, x_n) = 0 \end{Bmatrix}$$

Inicialmente como Newton es un método iterativo se comienza con un vector de aproximaciones iniciales de dimensión n como un punto de partida a las aproximaciones:

$$x^{(0)} = \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \\ \vdots \\ x_n^{(0)} \end{bmatrix} \qquad (2)$$

La fórmula del **método de Newton para sistemas de ecuaciones no lineales** es:

$$x^{(k)} = x^{(k-1)} - J\left(x^{(k-1)}\right)^{-1} F(x^{(k-1)}) \qquad (3)$$

donde,

$x^{(k)}$ son las aproximaciones a las soluciones en la k-ésima iteración y se puede representar como un vector columna de soluciones:

$$\begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \vdots \\ x_n^{(k)} \end{bmatrix}$$

$x^{(k-1)}$ como el vector de aproximación a las soluciones obtenido en la iteración $(k-1)$:

$$\begin{bmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \\ \vdots \\ x_n^{(k-1)} \end{bmatrix}$$

$J(x)$ como la matriz Jacobiana del sistema:

$$J(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x) & \frac{\partial f_1}{\partial x_2}(x) & \cdots & \frac{\partial f_1}{\partial x_n}(x) \\ \frac{\partial f_2}{\partial x_1}(x) & \frac{\partial f_2}{\partial x_2}(x) & \cdots & \frac{\partial f_2}{\partial x_n}(x) \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(x) & \frac{\partial f_n}{\partial x_2}(x) & \cdots & \frac{\partial f_n}{\partial x_n}(x) \end{bmatrix}$$

A $J\left(x^{(k-1)}\right)^{-1}$ como la inversa de la matriz Jacobiana evaluadas en $x^{(k-1)}$, y a $F(x^{(k-1)})$ como el vector de las funciones $f_1, f_2, \ldots, f_n$ evaluadas en $x^{(k-1)}$.

Las funciones $f_1, f_2, \ldots, f_n$ deben ser diferenciables y continuas en la región de estudio o aplicación, que la matriz Jacobiana no se anule en esa región, y que las condiciones iniciales y valores ya calculadas recientemente pertenezcan en esa región de estudio, ya que garantiza la convergencia del método a la solución.

La debilidad de Newton esta en el momento cuando se va a calcular e invertir la matriz Jacobiana $J(x)$ en cada paso. En la práctica, el cálculo explicito de la inversa de $J(x)$ se evita efectuando la operación en dos pasos. Primero, encontramos un vector y que satisfaga $J\left(x^{(k-1)}\right)y = -F\left(x^{(k-1)}\right)$, luego estaría

listo, para obtener la nueva aproximación $x^{(k)}$ agregando y a $x^{(k-1)}$. En el siguiente algoritmo se emplea este procedimiento de dos pasos que corresponden a los pasos 4 y 5.

ALGORITMO DEL MÉTODO DE NEWTON

Para aproximar la solución del sistema no lineal $F(x) = 0$ dada una aproximación inicial x:

ENTRADA numero de ecuaciones e incógnitas n; aproximación inicial $x = (x_1, \dots, x_n)^t$, tolerancia TOL; número máximo de iteraciones N.

SALIDA solución aproximada $x = (x_1, \dots, x_n)^t$ o un mensaje de que se rebaso el número de iteraciones.

Paso 1 Tome $k = 1$.

Paso 2 Mientras $(k \leq N)$ haga los pasos 3-7.

Paso 3 Calcule $F(x)$ y $J(x)$, donde $J(x)_{i,j} = (\partial f_i(x)/\partial x_j)$ para $1 \leq i, j \leq n$.

Paso 4 Resuelva el sistema lineal $n \times n$ $J(x)y = -F(x)$.

Paso 5 Tome $x = x + y$.

Paso 6 Si $\|y\| < TOL$, entonces SALIDA (x);
(Procedimiento terminado exitosamente.)
PARAR.

Paso 7 Tome $k = k + 1$

Paso 8 SALIDA ('Número máximo de iteraciones excedido');
(Procedimiento terminado sin éxito.)
PARAR.

EJEMPLO 4

Considere el siguiente sistema no lineal con dos incógnitas:

$$\begin{aligned} E_1&: \ 4x_1^2 - 20x_1 + \frac{1}{4}x_2^2 = -8, \\ E_2&: \ \frac{1}{2}x_1x_2^2 + 2x_1 - 5x_2 = -8. \end{aligned} \qquad (4)$$

Mediante el método de Newton para sistema no lineal halle la solución aproximada de (5), las aproximaciones iniciales dadas son $\left\{x_1^{(0)}, x_2^{(0)}\right\} = \{0,0\}^t$.

Solución:

Haciendo al sistema (4) de la forma $f(x) = 0$, resulta;

$$\begin{aligned} E_1&: \ 4x_1^2 - 20x_1 + \frac{1}{4}x_2^2 + 8 = 0, \\ E_2&: \ \frac{1}{2}x_1x_2^2 + 2x_1 - 5x_2 + 8 = 0. \end{aligned} \qquad (5)$$

El sistema (5) se puede representar de esta manera:

$$F(x_1, x_2) = \begin{Bmatrix} 4x_1^2 - 20x_1 + \frac{1}{4}x_2^2 + 8 \\ \frac{1}{2}x_1x_2^2 + 2x_1 - 5x_2 + 8 \end{Bmatrix} \qquad (6)$$

O es lo mismo decir, que $F(x_1, x_2) = [\, f_1(x_1, x_2), f_2(x_1, x_2)\,]^t$, donde

$f_1(x_1, x_2) = 4x_1^2 - 20x_1 + \frac{1}{4}x_2^2 + 8$ y $f_2(x_1, x_2) = \frac{1}{2}x_1x_2^2 + 2x_1 - 5x_2 + 8.$

Las derivadas parciales de las funciones f_1 y f_2 con respecto a cada variable son:

$$\frac{\partial f_1}{\partial x_1}(x) = 8x_1 - 20 \quad \text{y} \quad \frac{\partial f_1}{\partial x_2}(x) = \frac{1}{2}x_2^2$$

$$\frac{\partial f_2}{\partial x_1}(x) = \frac{1}{2}x_2^2 + 2 \quad \text{y} \quad \frac{\partial f_2}{\partial x_2}(x) = x_1x_2 - 5.$$

La Matriz Jacobiana para este sistema es:

$$J_F(x_1, x_2) = \begin{bmatrix} 8x_1 - 20 & \frac{1}{2}x_2 \\ \frac{1}{2}x_2^2 + 2 & x_1x_2 - 5 \end{bmatrix} \qquad (7)$$

La formula de Newton es:

$$x^{(k)} = x^{(k-1)} - J\left(x^{(k-1)}\right)^{-1} F(x^{(k-1)})$$

Para la primera iteración $x^{(1)}$, tenemos;

$$x^{(1)} = x^{(0)} - J\left(x^{(0)}\right)^{-1} F\left(x^{(0)}\right)$$

$$= \begin{bmatrix}0\\0\end{bmatrix} - \begin{bmatrix} 8x_1^{(0)} - 20 & \frac{1}{2}x_2{}^{(0)} \\ \frac{1}{2}x_2^{2(0)} + 2 & x_1{}^{(0)}x_2{}^{(0)} - 5 \end{bmatrix}^{-1} \begin{bmatrix} 4x_1^{2(0)} - 20x_1{}^{(0)} + \frac{1}{4}x_2^{2(0)} + 8 \\ \frac{1}{2}x_1^{(0)}x_2^{2(0)} + 2x_1{}^{(0)} - 5x_2{}^{(0)} + 8 \end{bmatrix}$$

$$= \begin{bmatrix}0\\0\end{bmatrix} - \begin{bmatrix}-20 & 0\\ 2 & -5\end{bmatrix}^{-1} \begin{bmatrix}8\\8\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix} - \begin{bmatrix}-0.05 & 0\\ -0.02 & -0.20\end{bmatrix} \begin{bmatrix}8\\8\end{bmatrix} = \begin{bmatrix}0.40\\1.76\end{bmatrix}$$

Las primeras iteraciones de $\begin{Bmatrix} x_1^{(1)} \\ x_2^{(1)} \end{Bmatrix}$ fueron $\begin{Bmatrix} 0.40 \\ 1.76 \end{Bmatrix}$. Para el resto de las iteraciones usaremos el software de MATLAB para más rapidez en la obtención de los resultados, la tabla 3 nos proporcionan dichos resultados con una tolerancia de 10^{-4} y con 4 iteraciones.

k	x_1	x_2	Error
1	0.400000	1.760000	1.804882
2	0.495894	1.983423	0.243133
3	0.499988	1.999937	0.017013
4	0.500000	2.000000	0.000064

Tabla 3

∴ Los valores aproximados de las variables del sistema no lineal son:

$$\begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} \rightarrow \begin{Bmatrix} 0.5 \\ 2.0 \end{Bmatrix}$$

En el anexo 4 se presenta el programa de Newton para un sistema de dos ecuaciones con dos variables implementado en MATLAB con la salida que corresponde a este ejemplo.

2.6 MÉTODO DE RUNGE-KUTTA DE CUARTO ORDEN

Para resolver un problema de ecuaciones diferenciales ordinarias con condiciones de valor inicial utilizamos el notorio método de un paso: **Runge-Kutta de cuarto orden** el cual proporciona un margen de error mínimo a la solución real y es considerado bastante preciso, estable y fácil de programar en un ordenador. Como el nombre lo indica, hay *métodos de Runge-Kutta de distintos órdenes.* Este método fue descubierto por los matemáticos alemanes *Carl D. T. Runge* y *Martin W. Kutta*.

El método de Runge-Kutta de cuarto orden se construye a partir de un método de Taylor de orden cuatro. A continuación, se presenta los detalles.

Se supone que la solución $y(t)$ del problema de valor inicial:

$$y' = f(t, y), \qquad a \leq t \leq b, \qquad y(a) = \alpha$$

tiene $(n+1)$ derivadas continuas. Al desarrollar la solución $y(t)$ en función de n-ésimo polinomio de Taylor alrededor de t_i y calculando en t_{i+1} se obtiene el siguiente resultado;

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \cdots + \frac{h^n}{n!}y^n(t_i) + \frac{h^{n+1}}{(n+1)!}y^{n+1}(\xi_i), \qquad (1)$$

para alguna ξ_i que se encuentra en (t_i, t_{i+1}).

La diferenciación sucesiva de la solución $y(t)$ resulta así:

$$\begin{array}{c} y'(t) = f\big(t, y(t)\big), \\ y''(t) = f'\big(t, y(t)\big), \\ \vdots \qquad\qquad \vdots \\ y^{(k)}(t) = f^{(k-1)}\big(t, y(t)\big) \end{array} \qquad (2)$$

Al sustituir (2) en la ecuación (1), lo que resulta es lo siguiente;

$$y(t_{i+1}) = y(t_i) + hf\big(t_i, y(t_i)\big) + \frac{h^2}{2}f'\big(t_i, y(t_i)\big) + \cdots + \frac{h^n}{n!}f^{(n-1)}\big(t_i, y(t_i)\big) + \frac{h^{n+1}}{(n+1)!}f^{(n)}\big(\xi_i, y(\xi_i)\big).$$

$$(3)$$

En consecuencia, la ecuación (3) recibe el nombre de *método de Taylor de orden* n y se obtiene anulando el término del residuo que lleva ξ_i. El *método de Taylor de orden* n se representa así:

$$w_0 = \alpha,$$

$$w_{i+1} = w_i + hT^{(n)}(t_i, w_i)$$

para cada $i = 0,1,2, \ldots, N-1$,

donde,

$$T^{(n)}(t_i, w_i) = f(t_i, w_i) + \frac{h^2}{2}f'(t_i, w_i) + \cdots + \frac{h^n}{n!}f^{(n-1)}(t_i, w_i)$$

Dado que los métodos de Taylor requieren la evaluación de las derivadas de $f(t, y)$, el método de las ecuaciones de diferencia que resulta de sustituir en el método de Taylor de orden cuatro las derivadas $f'(t, w)$, $f''(t, w)$ y $f'''(t, w)$ por expresiones que solo dependan de $f(t, w)$ es el llamado **método de Runge-Kutta de cuarto orden** y la fórmula es la siguiente:

$$w_0 = \alpha,$$
$$w_{i+1} = w_i + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4] \quad i = 0,1,..,N-1 \ (4)$$

donde k_1, k_2, k_3 y k_4 son de la forma,

$$k_1 = hf\ (t_i, w_i)\ ,$$
$$k_2 = hf\ (t_i + \frac{1}{2}h, w_i + \frac{1}{2}k_1)\ , \qquad (5)$$
$$k_3 = hf\ (t_i + \frac{1}{2}h, w_i + \frac{1}{2}k_2)\ ,$$
$$k_4 = hf\ (t_i + h, w_i + k_3)\ ,$$

El método Runge-Kutta de cuarto orden requiere realizar cuatro evaluaciones por paso, y es bastante preciso en aproximar a la solución exacta del problema de valor inicial. La única ventaja del método es obvia, que no hay que calcular las formulas de las derivadas de orden superior ni hay que incluirlas en el programa.

ALGORITMO DEL MÉTODO DE RUNGE-KUTTA DE CUARTO ORDEN

Para aproximar la solución del problema inicial

$y' = f(t, y), \quad a \le t \le b,\ y(a) = \alpha$ en $(N+1)$ números igualmente espaciado en el intervalo $[a, b]$:

ENTRADA: puntos extremos a, b; entero N; condición inicial α.

SALIDA: aproximación w de y en los $(N+1)$ valores de t.

Paso 1 Tomar $h = (b-a)N$

$$t = a;$$
$$w = \alpha;$$

SALIDA: (t, w).

Paso 2 para $i = 1,2, \ldots, N$ seguir los pasos 3-5

Paso 3 Tomar
$$K_1 = hf(t, w)$$
$$K_2 = hf\left(t + \frac{h}{2}, w + \frac{k_1}{2}\right);$$
$$K_3 = hf\left(t + \frac{h}{2}, w + \frac{k_2}{2}\right);$$
$$K_4 = hf(t + h, w + K_3);$$

Paso 4 tomar $w = w + (k_1 + 2k_2 + 2k_3 + k_4)/6$; (calcular w_i)

Paso 5 SALIDA (t, w)

Paso 6 PARAR.

EJEMPLO 5

Considere el problema de valor inicial:

$$y' = 1 + \frac{1}{t}y \qquad (6)$$

en el intervalo $1 \leq t \leq 2$, con una condición inicial de $y(1) = 2$, y con $h = 0.25$. Mediante el método de Runge-Kutta de cuarto orden halle las soluciones aproximadas de (6).

Solución:

Tomando en cuenta que la formula (4) depende de los valores de w_i, k_1, k_2, k_3 y k_4. Hallemos estos valores manteniendo el orden, es decir, como k_2 depende de k_1, hallemos k_1, para k_3 debemos tener el valor de k_2, y así sucesivamente.

Usando el paquete de formulas (5), calculemos para la primera aproximación w_1, como lo decía anteriormente.

Para k_1, es;

$$k_1 = hf\,(t_i, w_i) = (0.25)f(t_0, w_o) = (0.25)\left(1 + \frac{1}{(1)}(2)\right) = (0.25)(3) = 0.75$$

Para k_2 el resultado es;

$$k_2 = (0.25)f\left(t_0 + \frac{1}{2}(0.25), w_0 + \frac{1}{2}(0.75)\right) = (0.25)\left(1 + \frac{0.25}{2}, 2 + \frac{0.75}{2}\right)$$
$$= (0.25)(1.125, 2.375) = (0.25)\left(1 + \frac{1}{(1.125)}(2.375)\right) = 0.777777777$$

Para k_3 nos da;

$$k_3 = (0.25)f\left(t_0 + \tfrac{1}{2}(0.25), w_0 + \tfrac{1}{2}(0.777777777)\right) = (0.25)\left(1 + \tfrac{0.25}{2}, 2 + \tfrac{0.777777777}{2}\right) =$$
$$(0.25)(1.125, 2.388888889) = (0.25)\left(1 + \tfrac{1}{(1.125)}(2.388888889)\right) = 0.780864197$$

y para k_4 es;

$$k_4 = (0.25)(t_0 + h, w_0 + k_3) = (0.25)(1 + 0.25, 2 + 0.780864197)$$
$$= (0.25)(1.25, 2.780864197) = (0.25)\left(1 + \frac{1}{1.25}(2.780864197)\right)$$
$$= 0.806172839$$

Finalmente, se sustituye los valores de w_0, k_1, k_2, k_3 y k_4 en la ecuación (4) para hallar la primera aproximación w_1 de (6):

$$w_1 = w_0 + \frac{1}{6}[\, k_1 + 2k_2 + 2k_3 + k_4\,]$$
$$= 2 + \frac{1}{6}[0.75 + 2(0.777777777) + 2(0.780864197 + 0.806172839)]$$
$$= 2.778909465$$

Para los valores posteriores se calculan de manera igual. La tabla 4, resume los resultados del ejemplo usando el software de MATLAB; en la que se presenta las aproximaciones de la función solución en el intervalo dado $1 \leq t \leq 2$.

t	w
1.0	2.000000
1.3	2.778909
1.5	3.608165
1.8	4.479285
2.0	5.386243

Tabla 4

En el anexo 5 se presenta el programa de Runge-Kutta de cuarto orden implementado en MATLAB con la salida que corresponde a este ejemplo.

2.7 MÉTODO PREDICTOR-CORRECTOR DE ADAMS DE CUARTO ORDEN

El método de la sección anterior es un *método de un paso*, el que estudiaremos en esta sección es un *método de múltiple pasos*. Se conoce como *método de múltiple pasos* a aquellos que emplean la aproximación en más de un punto para determinar la aproximación en el siguiente punto. A continuación, definimos lo que es el método de múltiple pasos.

Definición 2.1 (Método multipasos de paso n)

Un método multipasos de paso n para resolver el problema de valor inicial de la forma:

$$y' = f(t,y), \qquad a \le t \le b, \qquad y(a) = \alpha,$$

es aquel cuya ecuación de diferencias para obtener la aproximación w_{i+1} en el punto t_{i+1} puede ser representada por medio de la siguiente ecuación, donde n es un entero mayor que 1:

$$w_{i+1} = a_{n-1}w_i + a_{n-2}w_{i-1} + \cdots + a_0 w_{i+1-n} + h[b_n f(t_{i+1}, w_{i+1}) + b_{n-1} f(t_i, w_i) + b_0 f(t_{i+1-n}, w_{i+1-n})] \qquad (1)$$

en donde $i = n-1, n, .., N-1$, y $h = (b-a)N$, las $a_0, a_1, \dots, a_{n-1}$ y los $b_0, b_1, \dots, b_n$ son constantes y los valores iniciales son determinados específicamente así:

$$\begin{aligned} w_0 &= \alpha, \\ w_1 &= \alpha_1, \\ w_2 &= \alpha_2, \\ &\vdots \\ w_{n-1} &= \alpha_{n-1} \end{aligned}$$

En el caso en que $b_n = 0$, el *método es explicito o abierto,* porque w_{i+1} resulta explícitamente de los valores previamente determinados y si $b_n \neq 0$, el *método es implícito o cerrado*, porque w_{i+1} se encuentra en ambos lados e implícitamente especificado de la ecuación (1).

A continuación, se definen dos métodos muy importantes de manera independiente, un *método explicito de cuatro pasos* llamado el *método de Adams-Bashforth de cuarto orden y* el método implícito de tres pasos denominado *método de Adams-Moulton de cuarto orden.*

Definición 2.2 (Método explicito de Adams-Bashforth de cuarto orden)

El método de Adams-Bashforth de cuarto orden está definido por las ecuaciones:

$$w_0 = \alpha, \quad w_1 = \alpha_1, \quad w_2 = \alpha_2, \quad w_3 = \alpha_3,$$

$$w_{i+1} = w_1 + \frac{h}{24}[55\, f(t_i, w_i) - 59\, f(t_{i-1}, w_{i-1}) + 37\, f(t_{i-2}, w_{i-2}) - 9\, f(t_{i-3}, w_{i-3})],$$

para cada $i = 3,4, \dots, N-1$.

(2)

Definición 2.3 (Método implícito de Adams-Moulton de cuarto orden)

El método de Adams-Moulton de cuarto orden está definido por las ecuaciones:

$$w_0 = \alpha, \quad w_1 = \alpha_1, \quad w_2 = \alpha_2,$$

$$w_{i+1} = w_1 + \frac{h}{24}[9\, f(t_{i+1}, w_{i+1}) + 19\, f(t_i, w_i) - 5\, f(t_{i-1}, w_{i-1}) + f(t_{i-2}, w_{i-2})],$$

para cada $i = 2,3, \dots, N-1$.

(3)

El método multipasos implícito sirve para dar mejores resultados a las aproximaciones obtenidas en forma explícita. El método que veremos, a continuación, tiene la combinación de un método implícito con uno explicito, es decir, como un *método predictor-corrector*. Donde el explicito estima una aproximación, mientras que el implícito mejora dicha estimación.

El método que se estudia en esta sección, precisamente es el resultado de la combinación de un método implícito y otro explicito, y se llama el **método predictor-corrector de Adams de cuarto orden.**

El primer paso consiste en calcular los valores iniciales w_0, w_1, w_2 y w_3 con el método *(*2). Para ello aplicaremos el método de *Runge-Kutta cuarto orden* de la sección 2.6. El siguiente paso consiste en calcular una aproximación $w_4^{(0)}$de $y(t_4)$ utilizando como predictor el método explicito (2), esto es:

$$w_4^{(0)} = w_3 + \frac{h}{24}[55\, f(t_3, w_3) - 59\, f(t_2, w_2) + 37\, f(t_1, w_1) - 9\, f(t_0, w_0)] \qquad (4)$$

Es conveniente insertar el valor de $w_4^{(0)}$ en el lado derecho del método (3) y obtenemos:

$$w_4^{(1)} = w_3 + \frac{h}{24}\left[9\, f\left(t_4, w_4^{(0)}\right) + 19\, f(t_3, w_3) - 5\, f(t_2, w_2) + f(t_1, w_1)\right] \qquad (5)$$

Cuando tengamos el valor de $w_4^{(1)}$ como aproximación a $y(t_4)$ se utiliza la técnica que consiste en usar como predictor el método (2) y como corrector el método (3) para hallar los valores de $w_5^{(0)}$ y $w_5^{(1)}$, que son las aproximaciones

inicial y final de $y(t_5)$, y así se va calculando sucesivamente para los otros valores de $y(t_n)$.

ALGORITMO DEL MÉTODO PREDICTOR-CORRECTOR DE ADAMS DE CUARTO ORDEN

Para aproximar la solución del problema del valor inicial

$$y' = f(t, y), \qquad a \leq t \leq b, \qquad y(a) = \alpha,$$

en $(N+1)$ numeros uniformemente espaciados en el intervalo $[a, b]$:

ENTRADA: extremos a, b; entero N; condición inicial α.

SALIDA: aproximación w a y en los $(N+1)$ valores de t.

Paso 1 Tome $h = (b-a)/N$;

$t_0 = a$;
$w_0 = \alpha$;

SALIDA (t_0, w_0)

Paso 2 Para $i = 1,2,3,$ haga pasos 3-5.

(Calcule valores usando el Método de Runge-Kutta.)

Paso 3 Tome $K_1 = hf(t_{i-1}, w_{i-1})$;
$K_2 = hf\left(t_{i-1} + \frac{h}{2}, w_{i-1} + K_1\right)$;
$K_3 = hf\left(t_{i-1} + \frac{h}{2}, w_{i-1} + K_2\right)$;
$K_4 = hf(t_{i-1} + h, w_{i-1}\, K_3)$;

Paso 4 Tome $w_i = w_{i-1} + (k_1 + 2k_2 + 2k_3 + k_4)/6$;
$t_i = a + ih$.

Paso 5 SALIDA (t_i, w_i).

Paso 6 Para $i = 4, \ldots, N$ haga pasos 7-10.

Paso 7 Tome $t = a + ih$;
$w = w_3 + h[55f(t_3, w_3) - 59f(t_2, w_2) + 37f(t_1, w_1)$
$-9f(t_0, w_0)]/24; \quad (Predice\ w_i.)$

$w = w_3 + h[9f(t, w) - 19f(t_3, w_3) - 5f(t_2, w_2)$
$+f(t_1, w_1)]/24; \quad (Corrige\ w_i.)$

Paso 8 SALIDA (t, w).

Paso 9 Para $j = 0,1,2$
tome $t_j = t_{j+1}$; (Prepare la siguiente iteración.)
$w_j = w_{j+1}$.

Paso 10 Tome $t_3 = t$;
$w_3 = w$.

Paso 11 PARAR.

EJEMPLO 6

Considere el problema de valor inicial:

$$y' = -5y + 5t^2 + 2t \qquad (6)$$

en el intervalo $0 \leq t \leq 1$ y con $h = 0.2$. La condición inicial dada es $y(0) = \frac{1}{2}$.

Aplique el algoritmo del método de Predictor-Corrector de Adams de cuarto orden para hallar la solución aproximada en dicho intervalo.

Solución:

Mediante el algoritmo de dicho método implementado en el software de MATLAB, tenemos los resultados de la ecuación diferencial (6) resumida en la tabla 5 con el número de subintervalos $N = 5$.

t_i	w_i
0.0	0.500000
0.2	0.228333
0.4	0.231458
0.6	0.387630
0.8	0.638872
1.0	0.991353

Tabla 5

En el anexo 6 se presenta el programa del método de Predictor-Corrector de Adams de cuarto orden implementado en MATLAB con la salida que corresponde a este ejemplo.

2.8 MÉTODO DE RUNGE-KUTTA PARA SISTEMAS DE ECUACIONES DIFERENCIALES DE PRIMER ORDEN

Cuando se tiene un modelo que es representado por una ecuación diferencial ordinaria de mayor orden es recomendable transformar dicha ecuación en un sistema de ecuaciones diferenciables de primer orden, sujeto a condiciones de valor inicial, ya que de esa manera es más fácil hallar la solución aproximada del problema.

Un *sistema de ecuaciones diferenciales de orden* n de problemas de valor inicial de primer orden se puede expresar como:

$$\begin{aligned} u_1'(t) &= \frac{du_1}{dt} = f_1(t, u_1, u_2, \dots, u_n), \\ u_2'(t) &= \frac{du_2}{dt} = f_2(t, u_1, u_2, \dots, u_n), \\ &\vdots \qquad\qquad \vdots \\ u_n'(t) &= \frac{du_n}{dt} = f_n(t, u_1, u_2, \dots, u_n), \end{aligned} \qquad (1)$$

en donde, $a \le t \le b$; con las siguientes condiciones iniciales dadas:

$$\begin{aligned} u_1(a) &= \alpha_1, \\ u_2(a) &= \alpha_2, \\ &\vdots \quad \vdots \\ u_n(a) &= \alpha_n. \end{aligned} \qquad (2)$$

La idea es encontrar n funciones $u_1,\ u_2, \dots, u_n$ que satisfagan tanto el sistema de ecuaciones diferenciales como todas sus condiciones iniciales.

La siguiente definición nos será útil para explicar sobre la existencia, y la unicidad de las soluciones de los sistemas de ecuaciones diferenciales.

Definición 2.4 (Condición de Lipschitz para funciones de varias variables)

Sea $f(t, y_1, y_2 \dots, y_n)$ definida en el conjunto $D = \{f(t, u_1, u_2, \dots, y_n) \mid a \le t \le b, -\infty \le u_i \le \infty,$ para cada $i = 1,2, \dots n\}$ satisface una **condición de Lipschitz** sobre D en las variables $u_1,\ u_2, \dots, u_n$ si existe una constante $L > 0$ con la siguiente propiedad de que,

$$|f(t, u_1, u_2, \dots, u_n) - \mathrm{f}(t, z_1, z, \dots, z_n)| \le L \sum_{j=1}^{n} |u_j - z_j|,$$

para todo $(t, u_1, u_2, \dots, u_n)$ y $(t, z_1, z_2, \dots, z_n)$ en D.

Aplicando el teorema 1.3 de valor medio del primer capítulo podemos demostrar que f y sus primeras derivadas parciales son continuas en D y que si,

$$\left|\frac{\partial f(t,u_1,u_2,\ldots,u_n),}{\partial u_i}\right| \leq L,$$

para cada $i = 1,2,\ldots,n$, y para todo $(t,u_1,u_2,\ldots,u_n)$ en D, f satisfara una *condición de Lipschitz* en D con su constante L.

El siguiente teorema establece las condiciones para la existencia y unicidad de las soluciones de un sistema de ecuaciones diferenciales de primer orden.

Teorema 2.1

Se supone que $D = \{f(t,u_1,u_2,\ldots,y_n) \mid a \leq t \leq b, -\infty \leq u_i \leq \infty,$ para cada $i = 1,2,\ldots,n\}$, y que $f_i(t,u_1,u_2,\ldots,y_n)$ para cada $i = 1,2,\ldots,n$ es continúa en D y que en ese instante satisface una condición de Lipschitz

El método que aplicaremos para resolver el sistema (1) es una generalización del **clásico método de Runge-Kutta de orden cuatro** de la sección 2.6, que está dado por:

$$\begin{aligned} w_0 &= \alpha \\ w_{i+1} &= w_i + \frac{1}{6}[\,k_1 + 2k_2 + 2k_3 + k_4\,]\ , \quad \text{para cada } i = 0,1,\ldots,N-1. \end{aligned} \qquad (3)$$

donde k_1, k_2, k_3 y k_4 son de la forma,

$$\begin{aligned} k_1 &= hf\,(t_i, w_i)\ , \\ k_2 &= hf\,(t_i + \frac{h}{2}, w_i + \frac{1}{2}k_1)\ , \\ k_3 &= hf\,(t_i + \frac{h}{2}, w_i + \frac{1}{2}k_2)\ , \\ k_4 &= hf\,(t_i + h, w_i + k_3). \end{aligned} \qquad (4)$$

La generalización del método Runge-Kutta de cuarto orden es la siguiente:

1. Se selecciona un entero $m > 0$ y usando $h = (b-a)/m$. La partición del intervalo $[a,b]$ en m subintervalos

$$t_j = a + jh \quad \text{para cada } j = 0,1,\ldots,m.$$

2. Se denota w_{ij} para una aproximación a $u_i(t_j)$ en cada $j = 0,1,\dots,m$ y para cada $i = 1,2,\dots,n$. O sea que, w_{ij} aproxima la i-ésima solución $u_i(t)$ de (1) en el j-ésimo punto t_j.

3. Se usan las condiciones iniciales como:

$$\begin{aligned} w_{1,0} &= \alpha_1, \\ w_{2,0} &= \alpha_2, \\ &\vdots \quad \vdots \\ w_{n,0} &= \alpha_n. \end{aligned}$$

4. Se calculan los $k_{1,i}, k_{2,i}, k_{3,i}$ y $k_{4,\mathrm{i}}$ de la siguiente manera:

$$k_{1,i} = hf_i\left(t_j, w_{1,j}, w_{2,j}, \dots, w_{n,j}\right), \text{ para cada } i = 1,2,\dots,n;$$

$$k_{2,i} = hf_i\left(t_j + \frac{h}{2}, w_{1,j} + \frac{1}{2}k_{1,1}, w_{2,j} + \frac{1}{2}k_{1,2}, \dots, w_{n,j} + \frac{1}{2}k_{1,n}\right),$$

para cada $i = 1,2,\dots,n$;

$$k_{3,i} = hf_i\left(t_j + \frac{h}{2}, w_{1,j} + \frac{1}{2}k_{2,1}, w_{2,j} + \frac{1}{2}k_{2,2}, \dots, w_{n,j} + \frac{1}{2}k_{2,n}\right),$$

para cada $i = 1,2,\dots,n$;

$$k_{4,i} = hf_i\left(t_j + h, w_{1,j} + k_{3,1}, w_{2,j} + k_{3,2}, \dots, w_{n,j} + k_{3,n}\right),$$

para cada $i = 1,2,\dots,n$;

5. Por último, calculando los $w_{1,j}, w_{2,j}, \dots, w_{n,j}$; se obtienen los $w_{1,j+1}, w_{2,j+1}, \dots, w_{n,j+1}$ mediante la siguiente fórmula:

$$w_{i,j+1} = w_{i,j} + \frac{1}{6}\left[k_{1,i} + 2k_{2,i} + 2k_{3,i} + k_{4,i}\right], \text{ para cada } i = 1,2,\dots,n.$$

Observando bien, se puede notar que para calcular $k_{2,i}$ se deberá calcular todos los valores $k_{1,1}, k_{1,2}, \dots, k_{1,n}$. En general, cada $k_{l,1}, k_{l,2}, \dots, k_{l,n}$ debe calcularse antes de cualquiera de las expresiones $k_{l+1,i}$.

ALGORITMO DE RUNGE-KUTTA PARA SISTEMA DE ECUACIONES DIFERENCIALES

Para encontrar la solución aproximada del sistema de n-ésimo orden de los problemas de valor inicial de primer orden

$$u_j' = f_j(t, u_1, u_2, \dots, u_n), \ a \le t \le b, \ \text{con } u_j(a) = \alpha_j,$$

Para $j = 1,2,\dots,n$ en $(m+1)$ numeros uniformemente espaciados en el intervalo $[a,b]$:

ENTRADA Extremos a,b; número de ecuaciones n, entero m, condiciones iniciales $\alpha_1, \alpha_2, \dots, \alpha_n$

SALIDA las aproximaciones w_j a u_j en los $(m+1)$ valores de t.

Paso 1 tomar $h = (b-a)/m$;

$$t = a.$$

Paso 2 Para $j = 1,2,\dots n$ tomar $w_j = \alpha_j$.

Paso 3 SALIDA $(t, w_1, , w_2, \dots, w_n)$.

Paso 4 Para $i = 1,2,\dots,m$ haga los pasos 5-11.

Paso 5 Para $j = 1,2,\dots,n$ tomar

$$k_{1,j} = hf_j(t, w_1, w_2, \dots, w_n).$$

Paso 6 Para $j = 1,2,\dots,n$ tomar

$$k_{2,j} = hf_j\left(t + \frac{h}{2}, w_1 + \frac{1}{2}k_{1,1}, w_2 + \frac{1}{2}k_{1,2}, \dots, w_n + \frac{1}{2}k_{1,n}\right).$$

Paso 7 Para $j = 1,2,\dots,n$ tomar

$$k_{3,j} = hf_j\left(t + \frac{h}{2}, w_1 + \frac{1}{2}k_{2,1}, w_2 + \frac{1}{2}k_{2,2}, \dots, w_n + \frac{1}{2}k_{2,n}\right).$$

Paso 8 Para $j = 1,2,\dots,n$ tomar

$$k_{4,j} = hf_j\left(t + h, w_1 + k_{3,1}, w_2 + k_{3,2}, \dots, w_n + k_{3,n}\right).$$

Paso 9 Para $j = 1,2,\dots,n$ tomar

$$w_j = w_j + \left(k_{1,j} + 2k_{2,j} + 2k_{3,j} + k_{4,j}\right)/6;$$

Paso 10 Tomar $t = a + ih$

Paso 11 SALIDA $(t, w_1, w_2, \dots, w_n)$.

Paso 12 PARAR.

EJEMPLO 7

Considere el siguiente sistema de ecuaciones diferenciales:

$$\begin{aligned} u_1' &= u_2, \\ u_2' &= -u_1 - 2e^t + 1, \\ u_3' &= -u_1 - e^t + 1. \end{aligned} \qquad (5)$$

en el intervalo $0 \leq t \leq 2$ con las condiciones iniciales dadas:

$$\begin{aligned} u_1(0) &= 1, \\ u_2(0) &= 0, \\ u_3(0) &= 1. \end{aligned} \qquad (6)$$

Mediante la generalización del método de Runge-Kutta para sistemas de ecuaciones diferenciales resuelva el sistema de ecuaciones diferenciales de primer orden (5).

Solución:

Usando $m = 4$, se tiene el tamaño de paso $h = 0.5$ para aproximar la solución del sistema (5) y las condiciones iniciales (6) se introduce como $w_{1,0} = 1$, $w_{2,0} = 0$ y $w_{3,0} = 1$.

Para las primeras aproximaciones $w_{11}w_{21}$ y w_{31}:

Calculamos $k_{1,i}$ con $i = 1,2,3.$ de la siguiente manera:

$$k_{1,1} = hf_1(t_0, w_{1,0}, w_{2,0}, w_{3,0}) = (0.5)[t_0, w_{2,0}] = (0.5)(0) = 0$$

$$k_{1,2} = hf_2(t_0, w_{1,0}, w_{2,0}, w_{3,0}) = (0.5)[-w_{1,0} - 2e^{t_0} + 1] = (0.5)[-1 - 2e^{(0)} + 1] = -1$$

$$k_{1,3} = hf_3(t_0, w_{1,0}, w_{2,0}, w_{3,0}) = (0.5)[-w_{1,0} - e^{t_0} + 1] = (0.5)[-1 - e^{(0)} + 1] = -0.5$$

Los $k_{2,i}$ con $i = 1,2,3.$ así:

$$k_{2,1} = hf_1\left[t_0 + \frac{h}{2}, w_{2,0} + \frac{1}{2}k_{1,2}\right] = (0.5)[\,0.25, -0.5\,] = (0.5)(-0.5) = -0.25$$

$$k_{2,2} = hf_2\left(t_0 + \frac{h}{2}, w_{1,0} + \frac{1}{2}k_{1,1}, w_{2,0} + \frac{1}{2}k_{1,2}, w_{3,0} + \frac{1}{2}k_{1,3},\right)$$
$$= (0.5)\left[t_0 + \frac{h}{2}, w_{1,0} + \frac{1}{2}k_{1,1}\right] = (0.5)[-1 - 2e^{0.25} + 1]$$
$$= -1.284025417$$

$$k_{2,3} = hf_3\left(t_0 + \frac{h}{2}, w_{1,0} + \frac{1}{2}k_{1,1}, w_{2,0} + \frac{1}{2}k_{1,2}, w_{3,0} + \frac{1}{2}k_{1,3}\right)$$
$$= (0.5)\left[t_0 + \frac{h}{2}, w_{1,0} + \frac{1}{2}k_{1,1}\right] = (0.5)[-1 - e^{0.25} + 1]$$
$$= -0.642012708$$

Los $k_{3,i}$ con $i = 1,2,3.$ de esta manera:

$$k_{3,1} = hf_1\left(t_0 + \frac{h}{2}, w_{2,0} + \frac{1}{2}k_{2,1}\right) = (0.5)\left[w_{2,0} + \frac{1}{2}k_{2,1}\right] = (0.5)[-0.125]$$
$$= -0.0625$$

$$k_{3,2} = hf_2\left(t_0 + \frac{h}{2}, w_{1,0} + \frac{1}{2}k_{2,2}\right) = (0.5)[-0.357987291 - 2e^{0.25} + 1]$$
$$= -0.963019062$$

$$k_{3,3} = hf_3\left(t_0 + \frac{h}{2}, w_{1,0} + \frac{1}{2}k_{2,2}\right) = (0.5)\left[[-0.357987291 - e^{0.25} + 1]\right]$$
$$= -0.321006353$$

y finalmente los $k_{4,i}$ con $i = 1,2,3.$ así:

$$k_{4,1} = hf_1(w_{2,0} + k_{3,2}) = (0.5)[-0.963019062] = -0.481509531$$

$$k_{4,2} = hf_2(t_0 + h, w_{1,0} + k_{3,1}) = (0.5)[-0.9375 - 2e^{0.5} + 1] = -1.617471271$$

$$k_{4,3} = hf_3(t_0 + h, w_{1,0} + k_{3,1}) = (0.5)\left[[-0.9375 - e^{0.5} + 1]\right] = -0.793110635$$

De tal manera que, las primeras iteraciones w_{11}, w_{21} y w_{31} son:

$$w_{1,1} = w_{1,0} + \frac{1}{6}\left[k_{1,1} + 2k_{2,1} + 2k_{3,1} + k_{4,1}\right] = 0.707871$$

$$w_{2,1} = w_{2,0} + \frac{1}{6}\left[k_{1,2} + 2k_{2,2} + 2k_{3,2} + k_{4,2}\right] = -1.249887$$

$$w_{3,1} = w_{3,0} + \frac{1}{6}\left[\, k_{1,3} + 2k_{2,3} + 2k_{3,3} + k_{4,3} \,\right] = \ 0.398849$$

Estos valores ya calculados recientemente corresponde a $t = 0.5$. De manera similar se obtienen los valores para los otros t restantes. En la tabla 6, se tiene resumido los resultados de la solución del sistema de ecuaciones diferenciales (6), usando la herramienta computacional de MATLAB.

t_i	w_{1i}	w_{2i}	w_{3i}
0.0	1.000000	0.000000	1.000000
0.5	0.707871	-1.249887	0.398849
1.0	-0.336918	-3.017642	-0.299323
1.5	-2.413327	-5.405233	-0.923469
2.0	-5.894790	-8.709705	-1.320512

Tabla 6

En el anexo 7 se presenta el programa del método Runge-Kutta para sistemas de ecuaciones diferenciales para un sistema particular de tres ecuaciones implementado en MATLAB con la salida que corresponde a este ejemplo.

El sistema de n ecuaciones diferenciales de primer orden de la forma:

$$\begin{aligned} u_1'(t) &= \frac{du_1}{dt} = f_1(t, u_1, u_2, \dots, u_n), \\ u_2'(t) &= \frac{du_2}{dt} = f_2(t, u_1, u_2, \dots, u_n), \\ &\vdots \qquad\qquad\qquad \vdots \\ u_n'(t) &= \frac{du_n}{dt} = f_n(t, u_1, u_2, \dots, u_n), \end{aligned} \qquad (8)$$

en donde, $a \le t \le b$; con las siguientes condiciones iniciales dadas:

$$\begin{aligned} u_1(a) &= \alpha_1, \\ u_2(a) &= \alpha_2, \\ &\vdots \qquad \vdots \\ u_n(a) &= \alpha_n, \end{aligned} \qquad (9)$$

está relacionado con el problema general de valor inicial de n-ésimo orden de la forma:

$$y^{(n)} = f\left(t, y, y', y'', \dots, y^{(n-1)}\right), \qquad (10)$$

en donde, $a \le t \le b$; con las siguientes condiciones iniciales dadas:

$$\begin{aligned} y(a) &= \alpha_1, \\ y'(a) &= \alpha_2, \\ &\vdots \qquad \vdots \\ y^{n-1}(a) &= \alpha_n. \end{aligned} \qquad (11)$$

Para resolver este problema general no se requiere nuevas técnicas sino que reetiquetamos las variables y reducimos la ecuación diferencial de orden superior a un sistema de ecuaciones diferenciales de primer orden, luego aplicamos el método de solución ya estudiado.

Un problema general de valor inicial (10) con sus respectivas condiciones de valores iniciales (11) puede convertirse en un sistema de la forma (8) con (9).

Sean $u_1(t) = y(t),\ u_2(t) = y'(t), \dots, u_n(t) = y^{(n-1)}(t)$ con esto se produce el sistema de ecuaciones diferenciales ordinarias de primer orden:

$$\begin{aligned} \frac{du_1}{dt} &= \frac{dy}{dt} = u_2, \\ \frac{du_2}{dt} &= \frac{dy'}{dt} = u_3, \\ &\vdots \quad \vdots \\ \frac{du_{n-1}}{dt} &= \frac{dy^{(n-2)}}{dt} = u_n, \end{aligned} \qquad (12)$$

y

$$\frac{du_n}{dt} = \frac{dy^{(n-1)}}{dt} = y^{(n)} = f\left(t, y, y', y'', \dots, y^{(n-1)}\right) = f(t, u_1, u_2, \dots, u_n).$$

Las condiciones iniciales son:

$$\begin{aligned} u_1(a) &= y(a) = \alpha_1, \\ u_2(a) &= y'(a) = \alpha_2, \\ &\vdots \quad \vdots \\ u_n(a) &= y^{(n-1)}(a) = \alpha_n. \end{aligned} \qquad (13)$$

EJEMPLO 8

Considere la ecuación de tercer orden:

$$y''' + 2y'' - y' - 2y = e^t \qquad (14)$$

en el intervalo $0 \le t \le 3$. Los valores iniciales son $y(0) = 1, y'(0) = 2$ y $y''(0) = 0$; y con $h = 0.2$.

Use el algoritmo del método de Runge-Kutta para sistemas de ecuaciones diferenciales para aproximar las soluciones de la ecuación (14).

Solución:

Haciendo $u_1(t) = y(t)$, $u_2(t) = y'(t)$ y $u_3(t) = y''(t)$ transformamos la ecuación superior (14) a un sistema:

$$u_1'(t) = u_2(t),$$

$$u_2'(t) = u_3(t),$$

$$u_3'(t) = -2u_3(t) + u_2(t) + 2u_1(t) + e^t.$$

Las condiciones iniciales transformadas resultan así,

$$u_1(0) = 1,\ u_2 = 2,\ \text{y}\ \ u_3(0) = 0.$$

Ahora, aplicando el algoritmo del método Runge-Kutta para sistemas de ecuaciones diferenciales se resuelve la ecuación (14), tomando el tamaño de paso $h = 0.2$, encontramos los valores para u_1, u_2 y u_3, mediante el software de de MATLAB, los resultados se podrán ver en la siguiente tabla 7.

t_i	w_{1i}	w_{2i}	w_{3i}
0.0	1.000000	2.000000	0.000000
0.2	1.406337	2.094395	0.919616
0.4	1.849181	2.361889	1.747229
0.6	2.361909	2.792901	2.567577
0.8	2.977556	3.393107	3.450121
1.0	3.731627	4.181249	4.457219
1.2	4.664604	5.188356	5.650332
1.4	5.824428	6.458085	7.095094
1.6	7.269132	8.048019	8.865841
1.8	9.069833	10.031843	11.050033
2.0	11.314246	12.502434	13.752964
2.2	14.110911	15.575943	17.103040
2.4	17.594350	19.397023	21.257977
2.6	21.931402	24.145402	26.412221
2.8	27.329028	30.044109	32.805972
3.0	34.043957	37.369687	40.736233

Tabla 7

En el anexo 8 se presenta el programa del método de Runge-Kutta para sistemas de ecuaciones diferenciales en particular para un sistema de tres ecuaciones implementado en MATLAB con la salida que corresponde a ejemplo.

En el anexo 9 se presenta una tabla comparativa de los métodos numéricos estudiados en el este capítulo 2.

CAPITULO 3
APLICACIONES

3.1 INTRODUCCION

En este capítulo veremos algunas aplicaciones de los métodos numéricos estudiados en el segundo capítulo y que se presentan en ciertos campos de la ciencia e ingeniería.

El primer caso, presenta una aplicación del método de Newton-Raphson en el campo de las Finanzas. El segundo caso, hace referencia a una aplicación de los métodos de Eliminación Gaussiana con Sustitución hacia Atrás y de Gauss-Seidel en el campo de la ingeniería civil.

El tercer caso, aplica el método de Newton en el campo de la ingeniería ambiental y el cuarto caso, aplica el método de Runge-Kutta de cuarto orden, el método de Predictor-Corrector de Adams de cuarto orden y el método de Runge-Kutta para sistema de ecuaciones diferenciales en ciencias demográficos y de la salud.

3.2 CASO ESPECIAL PARA ECUACIONES DE UNA VARIABLE

Como se indicaba en el capítulo 2, el método de Newton-Raphson nos ayuda a resolver ecuaciones de una variable de la forma $f(x) = 0$, presentamos a continuación una aplicación de este método en el campo de las Finanzas.

ANUALIDADES ORDINARIAS

Una anualidad es un valor fijo de dinero que se paga o recibe a intervalos iguales de tiempo. Una anualidad ordinaria o vencida es aquella en que el pago se hace al final de cada periodo de interés. Una anualidad de este tipo es el más común, ya que los pagos se hacen al final de periodo y pueden ocurrir en días, semanas, quincenas, meses, trimestrales, etc.

Para los problemas relativos al dinero necesario para pagar una hipoteca de una casa durante un periodo fijo de tiempo se utiliza la formula:

$$P = \frac{A}{i}[1 - (1 + i)^{-n}] \qquad (1)$$

conocida como la ecuación del *valor presente de una anualidad ordinaria,* en donde P es el importe de la hipoteca, A es el importe de cada pago e i es la tasa de interés por periodo para n periodos.

Ahora, supongamos que se necesita una hipoteca de $135,000 por una casa a un plazo de 30 años y que los pagos máximos que puede realizar son de $1000 mensuales. ¿Cuál será el interés más alto que podrá pagar?

Solución del problema:

La formula (1) se puede expresar en términos de una función que solo dependa de la tasa de interés i:

$$f(i) = -P + \frac{A}{i}[1 - (1+i)^{-n}] \qquad (2)$$

La derivada de (2) esta dada por,

$$f'(i) = \frac{A}{i}[-\frac{1}{i}(1 - (1+i)^{-n}) + n(1+i)^{-n-1}] \qquad (3)$$

El método de Newton-Raphson de la sección 2.2, para este problema se representa así:

$$i_k = i_{k-1} - \frac{f(i_{k-1})}{f'(i_{k-1})} \quad k \geq 1 \quad (4)$$

en donde i_k son las k-ésima aproximaciones a la tasa de interés deseada. Para que converja rápidamente a la raíz necesitamos de un buen punto inicial. Así, $i_0 = 0.01$ es nuestro punto inicial.

Sustituyendo rápidamente $P = \$135{,}000$, $A = \$1000$ mensuales, nuestro $n = 30$ años se convierte en (30 años) (12 meses)=360 meses el resultado es;

$$f(i) = -135{,}000 + \frac{1000}{i}[1 - (1+i)^{-360}]$$

y (5)

$$f'(i) = \frac{1000}{i}[\frac{1}{i}(1 - (1+i)^{-360}) - (360)(1+i)^{-361}]$$

Luego, sustituyendo (5) en (4), tenemos,

$$i_k = i_{k-1} - \frac{-135{,}000 + \frac{1000}{i_{k-1}}[1 - (1+i_{k-1})^{-360}]}{\frac{1000}{i_{k-1}}[-\frac{1}{i_{k-1}}(1 - (1+i)^{-360}) + (360)(1+i_{k-1})^{-361}]}$$

Anteriormente, decíamos que nuestro punto inicial es $i_0 = 0.01$. Entonces, haciendo el uso del software de MATLAB obtenemos las k-aproximaciones del problema con una tolerancia de 10^{-4} y en tan solo 4 iteraciones la tasa de interés más alto que podrá pagar durante 30 años si la anualidad es de $1000 mensuales es $i = 0.00674992$. En la tabla 8 muestra los resultados obtenidos mediante el uso del software MATLAB.

k	i_k	$i_k - i_{k-1}$
1	0.005672	0.004328
2	0.006646	0.000973
3	0.006749	0.000103
4	0.006750	0.000001

Tabla 8

En el anexo 10 se muestra el programa de Newton-Raphson implementado para el caso de esta sección con su salida correspondiente.

3.3 CASO ESPECIAL PARA SISTEMAS DE ECUACIONES LINEALES

El propósito de esta sección está basado en la aplicación de dos métodos para resolver sistemas de ecuaciones lineales de la forma $Ax = b$, el primero es el método directo de Eliminación Gaussiana con Sustitución hacia Atrás de la sección 2.3 y el segundo es el método iterativo de Gauss-Seidel de la sección 2.4. Haremos uso de estos dos métodos para hallar la solución del problema considerado a continuación.

ÁREA DE CONSTRUCCIONES

En ingeniería civil surge problemas de construcciones de edificios en donde no pueden obtener con exactitud las cantidades de los materiales que van a usar para construir dichos edificios por lo que buscan métodos que aproximen las cantidades a usar.

Un ingeniero civil que trabaja en la construcción de un edificio de una planta, requieren de las cantidades de 4800 de arena, 5800 de grava fina y 5700 de grava gruesa, las tres cantidades en términos de m^3, respectivamente, para cierto proyecto de construcción. Hay tres canteras de las que puede obtenerse dichos materiales. En la tabla 9, muestra la composición de dichos materiales en las canteras.

	Arena %	Grava Fina %	Grava Gruesa %
Cantera 1	52	30	18
Cantera 2	20	50	30
Cantera 3	25	20	55

Tabla 9

Lo que el ingeniero desea saber es lo siguiente:

¿Cuántos m^3 debe extraerse de cada cantera a fin de satisfacer las necesidades?

Solución del problema:

Sean x_1, x_2 y x_3 las cantidades (en m^3) en cada uno de las canteras que hay que usar para formar una cantidad total de $16300\ m^3$ de mezcla que contenga $b_1 = 4800, b_2 = 5800$ y $b_3 = 5700\ m^3$ de arena, grava fina y grava gruesa, respectivamente. El sistema de ecuaciones lineales para los ingredientes es:

$$\begin{aligned} E_1&:\ 0.52x_1 + 0.20x_2 + 0.25x_3 = 4800, \\ E_2&:\ 0.30x_1 + 0.50x_2 + 0.20x_3 = 5800, \\ E_3&:\ 0.18x_1 + 0.30x_2 + 0.55x_3 = 5700. \end{aligned} \qquad (1)$$

La matriz asociada al sistema (1) es:

$$A = \left[\begin{array}{cccc} 0.52 & 0.20 & 0.25 & 4800 \\ 0.30 & 0.50 & 0.20 & 5800 \\ 0.18 & 0.30 & 0.55 & 5700 \end{array}\right] \qquad (2)$$

Resolviéndolo (2) de manera directa usando el método Eliminación Gaussiana con Sustitución hacia Atrás, obtenemos,

$$\begin{aligned} x_1 &= 4005.813953 \\ x_2 &= 7131.395349 \\ x_3 &= 5162.790698 \end{aligned}$$

Resolviendo (2) de manera iterativa usando el método de Gauss-Seidel implementado en el software de MATLAB con una tolerancia de 10^{-4} , y hasta 10 iteraciones la solución del problema esta dado en la tabla 10.

k	x_1	x_2	x_3
1	9230.769231	6061.538462	4036.363636
2	4958.848843	7010.145239	4917.024793
3	4170.605296	7130.826905	5109.169046
4	4031.812226	7137.245046	5151.091428
5	4009.188719	7134.050197	5160.238130
6	4006.020054	7132.292716	5162.233774
7	4005.736564	7131.664552	5162.669187
8	4005.768832	7131.471026	5162.764186
9	4005.797593	7131.415770	5162.784913
10	4005.808880	7131.400707	5162.789436

Tabla 10

Así las soluciones aproximadas de las variables del sistema lineal son:

$$x_1 = 4005.808880$$
$$x_2 = 7131.400707$$
$$x_3 = 5162.789436$$

Comparando los resultados que nos ofrecieron los dos métodos, tenemos una mínima diferencia, ya que uno genero directamente y la otra de manera iterativa. El hecho de resolver con Gauss-Seidel una matriz de dimensión 3×3 no significa que es muy eficiente para resolver matrices de esta dimensión, puede haber decaimiento en otros casos. En este caso nos resulto excelente el uso de Gauss-Seidel por supuesto que, el método Eliminación Gaussiana con Sustitución hacia Atrás también nos ofreció buenos resultados y de manera directa.

En el anexo 11 y en el anexo 12 se muestran los programas de Eliminación Gaussiana con Sustitución hacia Atrás y el de Gauss-Seidel implementados para el caso de esta sección con sus salidas correspondientes.

3.4 CASO ESPECIAL PARA SISTEMAS DE ECUACIONES NO LINEALES

En esta sección se expondrá la aplicación del método de Newton para sistemas de ecuaciones no lineales de la forma $F(x) = 0$. El problema que se plantea se da en ingeniería ambiental.

Considere una tubería de sección circular que va del punto P_1 al punto P_2 que se divide en dos ramas, una que va al punto P_3 y la otra al punto P_4. Designando por Q al caudal que va de P_1 a P_2, por Q_1 al que va de P_2 a P_3, por Q_2 al que va de P_2 a P_4 y p_1, p_2, p_3 y p_4 a las presiones en los puntos P_1, P_2, P_3 y

P_4 respectivamente, caídas de presión en cada tramo y los caudales que por ellos circulan se pueden relacionar mediantes las siguientes ecuaciones:

$$\begin{Bmatrix} p_1 - p_2 = K_1 \cdot Q^{1.75} \\ p_2 - p_3 = K_2 \cdot Q_1^{1.75} \\ p_2 - p_4 = K_3 \cdot Q_2^{1.75} \\ Q = Q_1 + Q_2 \end{Bmatrix} \qquad (1)$$

Para un fluido y tubería concretos se han estimado los siguientes valores:

$$K_1 = 2.35 \cdot e^{-3}, K_2 = 4.67 \cdot e^{-3}, K_3 = 3.72 \cdot e^{-2}, \qquad (2)$$

y

$$p_1 = 75\ psi, p_3 = 20\ psi, p_4 = 15\ psi. \qquad (3)$$

Se desea estimar la presión p_2 existente en el punto P_2 así como las caudales Q, Q_1 y Q_2 que circulan por cada una de las ramas de la red de tuberías antes descrita.

Solución del problema:

El sistema de ecuaciones no lineales de acuerdo, a los datos anteriores (2) y (3) sustituyéndolos en (1) nos resulta,

$$\begin{matrix} E_1: 75 - p_2 - 2.35 \cdot e^{-3} \cdot Q^{1.75} = 0, \\ E_2: p_2 - 20 - 4.67 \cdot e^{-3} \cdot Q_1^{1.75} = 0, \\ E_3: p_2 - 15 - 3.72 \cdot e^{-2}, \cdot Q_2^{1.75} = 0, \\ E_4: Q - Q_1 + Q_2 = 0. \end{matrix} \qquad (4)$$

Haciendo $Q = Q_1 + Q_2$ de la ecuación E_4 y sustituyendo en la ecuación E_1 del sistema cuatro ecuaciones no lineales (4), el sistema se convierte en un sistema de tres ecuaciones no lineales:

$$\begin{matrix} E_1: 75 - p_2 - 2.35 \cdot e^{-3} \cdot (Q_1 + Q_2)^{1.75} = 0, \\ E_2: p_2 - 20 - 4.67 \cdot e^{-3} \cdot Q_1^{1.75} = 0, \\ E_3: p_2 - 15 - 3.72 \cdot e^{-2} \cdot Q_2^{1.75} = 0. \end{matrix} \qquad (5)$$

Ahora, representemos nuestro sistema (5) en función de p_2, Q_1 y Q_2 como sigue,

$$F(p_2, Q_1, Q_2) = \begin{Bmatrix} 75 - p_2 - 2.35 \cdot e^{-3} \cdot (Q_1 + Q_2)^{1.75} \\ p_2 - 20 - 4.67 \cdot e^{-3} \cdot Q_1^{1.75} \\ p_2 - 15 - 3.72 \cdot e^{-2} \cdot Q_2^{1.75} \end{Bmatrix} \qquad (5)$$

La matriz Jacobiana de (6) es:

$$J_F(p_2, Q_1, Q_2) = \begin{bmatrix} -1 & -0.204749318 \cdot (Q_1 + Q_2)^{0.75} & -0.204749318 \cdot (Q_1 + Q_2)^{0.75} \\ 1 & -0.406884816 Q_1^{0.75} & 0 \\ 1 & 0 & -0.881032693 Q_2^{0.75} \end{bmatrix}$$

Para aplicar el algoritmo del método de Newton necesitamos las condiciones de valores iniciales, como el punto P_2 esta entre P_1 y los valores P_3 y P_4 tomaremos a la presión intermedia p_2 como $p_2 = 50\ psi$. Para los caudales Q_1 y Q_2tomaremos $Q_1 = 16$ y $Q_2 = 7$, respectivamente.

Usando el software de MATLAB tenemos los resultados correspondientes a este problema con una tolerancia de 10^{-4} y con 10 iteraciones.

$$\begin{Bmatrix} p_1 \\ Q_1 \\ Q_2 \end{Bmatrix} \rightarrow \begin{Bmatrix} 43.959652 \\ 14.135547 \\ 10.130304 \end{Bmatrix}$$

Para el caso, de $Q_1 + Q_2$, esto es igual a $14.135547 + 10.130304 = 24.2659$.

En el anexo 13 se muestra el programa de Newton implementado para el caso de esta sección con su salida correspondiente.

3.5 CASO ESPECIAL PARA ECUACIONES Y SISTEMAS DE ECUACIONES DIFERENCIALES

En esta última sección de este capítulo tendremos la aplicación de las secciones 2.6, 2.7 y 2.8 del segundo capítulo para resolver el problema de estudio debido a que tiene la forma de ecuaciones diferenciales ordinarias. Las aplicaciones en este caso serán en ciencias demográficas y de la salud.

1. MODELO DE CRECIMIENTO POBLACIONAL

Un modelo sencillo para el crecimiento de una población que incorpora la suposición de que la tasa de cambio de la población p es proporcional a la que existe en cualquier tiempo t es:

$$\frac{dp}{dt} = Gp \qquad (1)$$

donde G es la tasa de crecimiento. Este modelo funciona adecuadamente cuando la población es ilimitada, falla ante la existencia de factores tales como

la falta de comida, contaminación y falta de espacio, los cuales inhibe el crecimiento. En tales casos, la tasa de crecimiento se considera que es inversamente proporcional a la población. Un modelo de esta relación es:

$$G = G'(p_{max} - p) \qquad (2)$$

donde G' es la tasa de crecimiento dependiente de la población (por persona-año) y p_{max} es la población máxima sostenible. Así, cuando la población es pequeña $(p \ll p_{max})$, la tasa de crecimiento será elevada y constante de $G'p_{max}$. En tales casos, el crecimiento es ilimitado y la ecuación (2) es idéntica a la (1). Sin embargo, conforme la población crece (es decir, es cero). Así, el modelo predice que, cuando la población alcanza el nivel máximo sostenible, el crecimiento es inexistente y el sistema se encontrara en estado sostenible. Al sustituir la ecuación (2) a la (1) se llega a lo siguiente:

$$\frac{dp}{dt} = G'(p_{max} - p)p \qquad (3)$$

Para nuestro interés usaremos la ecuación (3) para predecir la población de una ciudad en el tiempo $t = 30\ años$ con un tamaño de paso de 0.5 años, $G' = 10^{-5}$ por persona-año y $p_{max} = 20{,}000$ personas. Al tiempo $t = 0$, la ciudad tiene una población de 6,000 personas.

Solución del problema:

Utilizando los métodos de las secciones 2.6 y 2.7 del segundo capítulo, Runge-Kutta de cuarto orden y Predictor-Corrector de cuarto orden respectivamente, resolveremos el problema. Dado ya los parámetros al sustituirlos nos resulta,

$$\frac{dp}{dt} = 10^{-5}(20000 - p)p \qquad (4)$$

con un paso de 0.5 anual. Cuando $t = 0$, la ciudad tiene una población de 6,000 personas inicialmente. La pregunta es, ¿cuánto habrá crecido en $t = 30$ años?

Por medio del algoritmo del método Runge-Kutta de cuarto orden los resultados obtenidos con MATLAB están en la tabla 11, en donde el número de pasos M es 60.

t	w
0.0	6000.000000
0.5	6428.207310
1.0	6871.974217
1.5	7329.830900
2.0	7800.068110
2.5	8280.756475
3.0	8769.773704
3.5	9264.839118
4.0	9763.554437
4.5	10263.449358
5.0	10762.030135
5.5	11256.829135
6.0	11745.453302
6.5	12225.629526
7.0	12695.245120
7.5	13152.381980
8.0	13595.343378
8.5	14022.672822
9.0	14433.164879
9.5	14825.86826
10.0	15200.081843
10.5	15555.344526
11.0	15891.420081
11.5	16208.278113
12.0	16506.072364
12.5	16785.117442
13.0	17045.864982
13.5	17288.880079
14.0	17514.818668
14.5	17724.406376
15.0	17918.419190
15.5	18097.666172
16.0	18262.974309
16.5	18415.175512
17.0	18555.095688
17.5	18683.545759
18.0	18801.314474
18.5	18909.162825
19.0	19007.819886
19.5	19097.979881
20.0	19180.300296
20.5	19255.400870
21.0	19323.863313
21.5	19386.231598
22.0	19443.012722
22.5	19494.677817
23.0	19541.663527
23.5	19584.373568
24.0	19623.180422
24.5	19658.427093
25.0	19690.428898
25.5	19719.475253
26.0	19745.831433
26.5	19769.740283
27.0	19791.423857
27.5	19811.085000
28.0	19828.908834
28.5	19845.064168
29.0	19859.704824
29.5	19872.970867
30.0	19884.989762

Tabla 11

Ahora, Por medio del algoritmo del método Predictor-Corrector de Adams de cuarto orden los resultados obtenidos con MATLAB están en la tabla 12, en donde el número de pasos M es 60.

t_i	w_i
0.0	6000.000000
0.5	6428.207310
1.0	6871.974217
1.5	7329.830900
2.0	7800.068662
2.5	8280.757787
3.0	8769.775972
3.5	9264.842514
4.0	9763.559093
4.5	10263.455355
5.0	10762.037489
5.5	11256.837794
6.0	11745.463148
6.5	12225.640376
7.0	12695.256741
7.5	13152.394101
8.0	13595.355706
8.5	14022.685063
9.0	14433.176755
9.5	14825.879524
10.0	15200.092282
10.5	15555.353982
11.0	15891.428445
11.5	16208.285327
12.0	16506.078417
12.5	16785.122365
13.0	17045.868839
13.5	17288.882960
14.0	17514.820680
14.5	17724.407636
15.0	17918.419819
15.5	18097.666289
16.0	18262.974025
16.5	18415.174930
17.0	18555.094900
17.5	18683.544844
18.0	18801.313499
18.5	18909.161844
19.0	19007.818943
19.5	19097.979008
20.0	19180.299517
20.5	19255.400202
21.0	19323.862764
21.5	19386.231173
22.0	19443.012419
22.5	19494.677633
23.0	19541.663455
23.5	19584.373601
24.0	19623.180550
24.5	19658.427306
25.0	19690.429186
25.5	19719.475605
26.0	19745.831841
26.5	19769.740735
27.0	19791.424346
27.5	19811.085518
28.0	19828.909372
28.5	19845.064721
29.0	19859.705385
29.5	19872.971431
30.0	19884.990325

Tabla 12

Comparando los resultados obtenidos del método de un paso de Runge-Kutta de orden 4 en la tabla 11 y del método de múltiples pasos en la taba 12 la diferencia es mínima, ya que con Runge-Kutta de orden 4 fue de 19884.989762 años y con el Predictor-Corrector de Adams de orden 4 de 19884.990325 años, los dos durante los 30 años.

2. MODELO MATEMATICO PARA UNA EPIDEMIA

Otro de los modelos de esta sección, en este caso que, tiene la forma de un sistema de ecuaciones diferenciales ordinarias es conocido como el modelo matemático de epidemias abreviado **SIR** creado por *W. Kermack* y *A. McKendrick* en el año 1927. Este modelo consiste en predecir el comportamiento de agentes infecciosos que probablemente son muy dañinos para las poblaciones humanas o animales.

El modelo **SIR** considera a lo largo de un cierto tiempo t una enfermedad que se desarrolla únicamente en tres clases de individuos. La representación esquemática del progreso de la enfermedad que se da de un grupo a otro es:

$$S \to I \to R.$$

Dada una población fija $N = S + I + R$, el modelo matemático para una epidemia que afecta a una cierta población se representa así:

$$\begin{aligned} \frac{dS}{dt} &= -aSI \\ \frac{dI}{dt} &= aSI - rI \qquad (1) \\ \frac{dR}{dt} &= rI \end{aligned}$$

donde S representa a los individuos susceptibles que anteriormente no ha tenido enfermedad y que pueden resultar infectados al entrar en contacto con la enfermedad, I representa a los individuos infectados que pueden transmitir la enfermedad al grupo S, R representa los individuos recuperados de la enfermedad, y posteriormente, no podrán volver a enfermarse ni de transmitir la enfermedad a otros, a como la tasa de infección y por último, r como la tasa de recuperación.

Ahora, supongamos que una ciudad tiene 1000 habitantes, de los cuales 900 son susceptibles, 100 son infectados y no hay recuperados en el tiempo $t = 0$, calcule la progresión de la epidemia hasta que el número de individuos infectados caiga por debajo de 10.

Use los siguientes parámetros: $a = 0.002$ como la tasa de infección por semana y $r = 0.15$ como la tasa de recuperación por semana.

Solución del problema:

El método Runge-Kutta para sistemas de ecuaciones diferenciales de la sección 2.8 es el método que utilizaremos para resolver el problema propuesto.

Tomando las condiciones iniciales como $S(0) = 0.9$, $I(0) = 0.1$ y a $R(0) = 0$. Sustituyendo los parámetros de las tasas de infección y recuperación para saber la progresión de la epidemia en 35 semanas tenemos:

$$\frac{dS}{dt} = -(0.002)SI$$

$$\frac{dI}{dt} = (0.002)SI - (0.15)I$$

$$\frac{dR}{dt} = (0.15)I$$

Usando el software de MATLAB tenemos los siguientes resultados en la tabla 13.

t_i	w_{1i}	w_{2i}	w_{3i}
0.0	0.900000	0.100000	0.000000
1.0	0.899833	0.086226	0.013941
2.0	0.899689	0.074349	0.025962
3.0	0.899564	0.064108	0.036328
4.0	0.899457	0.055278	0.045265
5.0	0.899365	0.047664	0.052972
6.0	0.899285	0.041098	0.059617
7.0	0.899216	0.035437	0.065346
8.0	0.899157	0.030556	0.070287
9.0	0.899106	0.026347	0.074547
10.0	0.899062	0.022718	0.078220
11.0	0.899024	0.019589	0.081387
12.0	0.898991	0.016891	0.084118
13.0	0.898963	0.014564	0.086473
14.0	0.898939	0.012558	0.088503
15.0	0.898918	0.010828	0.090254
16.0	0.898900	0.009337	0.091764
17.0	0.898884	0.008051	0.093065
18.0	0.898870	0.006942	0.094188
19.0	0.898859	0.005986	0.095156
20.0	0.898849	0.005161	0.095990
21.0	0.898840	0.004450	0.096710
22.0	0.898833	0.003837	0.097330
23.0	0.898826	0.003309	0.097865
24.0	0.898821	0.002853	0.098326
25.0	0.898816	0.002460	0.098724
26.0	0.898812	0.002121	0.099067
27.0	0.898808	0.001829	0.099363
28.0	0.898805	0.001577	0.099618
29.0	0.898803	0.001360	0.099837
30.0	0.898800	0.001172	0.100027
31.0	0.898799	0.001011	0.100190
32.0	0.898797	0.000872	0.100331
33.0	0.898795	0.000752	0.100453
34.0	0.898794	0.000648	0.100558
35.0	0.898793	0.000559	0.100648

Tabla 13

Si observamos los resultados de la tabla 13 lo que se determina en la segunda columna es la velocidad con que la epidemia disminuye durante las 35 semanas. Note que en la semana 16 y en las siguientes, el número de infectados es ya menor de 10.

Cuando la infección se va propagando, la cantidad de susceptibles disminuye, de manera que a la infección se hace cada vez más difícil de propagarse, y empieza a decaer, ya que necesita individuos susceptibles para seguir con la propagación. Inevitablemente, algunos susceptibles pueden resultar libres de la infección. Otros pueden morirse a causa de gravedad de la infección.

Existen varias enfermedades que no se transmiten directamente de uno que está infectado a uno que es susceptible, sino que lo hacen por medio de mosquitos, amebas, roedores, entre otros animales. El estudio de tales enfermedades requerirá por tanto el análisis de dos poblaciones diferentes y en este caso existen términos como subpoblaciones dentro de la misma población infectada. En cambio, en este modelo sencillo solo se toman en cuenta dos procesos el que va del grupo susceptible al de infectado, de infectado a los recuperados sin importar los factores anteriores.

En el anexo 14 y en el anexo 15 se muestran los programas de Runge-Kutta de cuarto orden y el de Predictor-Corrector de Adams de cuarto orden implementados para el caso 1 de esta sección con sus salidas correspondientes.

En el anexo 16 se muestra el programa de Runge-Kutta para sistemas de ecuaciones diferenciales de primer implementado para el caso 2 de esta sección con su salida correspondiente.

CONCLUSIONES

Podemos destacar del análisis de cada método estudiado lo siguiente:

- El método de Newton-Raphson determina rápidamente una raíz de la ecuación $f(x) = 0$ siempre y cuando se comience con un buen punto inicial. La debilidad del método se tiene cuando la derivada de la función con la que se va a trabajar resulta muy compleja.

- Los métodos de Eliminación Gaussiana con Sustitución hacia Atrás y Gauss-Seidel ambos resuelven sistemas de ecuaciones lineales de la forma $Ax = b$ de dimensión nxn, el primero de manera directa y el segundo de manera iterada. El método de Eliminación Gaussiana con sustitución hacia atrás conlleva bastantes operaciones aritméticas y el error de redondeo puede resultar muy grande, aunque este método es muy bueno para hallar la solución de un sistema lineal de dimensión pequeña.

- El método de Gauss-Seidel es eficiente en el caso de hallar la solución de un sistema de ecuaciones de dimensión grande con la matriz de coeficientes del sistema estrictamente dominante diagonalmente. Rara vez se usa para resolver sistemas de dimensión pequeña, ya que el tiempo necesario para conseguir una exactitud satisfactoria rebasa el que requiere el método de Eliminación Gaussiana con Sustitución hacia Atrás.

- El método de Newton resuelve sistemas de ecuaciones no lineales del tipo $F(x) = 0$. El método da una convergencia cuadrática a las raíces del sistema no lineal cuando se conoce un valor inicial suficientemente preciso a la solución y cuando existe la inversa de la matriz Jacobiana.

- Los métodos de Runge-Kutta de cuarto orden y Predictor-Corrector de Adams de cuarto orden ambos resuelven ecuaciones diferenciales ordinarias de la forma $y' = f(t, y)$ con condición de valor inicial. El primero mediante un paso y el otro mediante múltiples pasos. El método de Runge-Kutta aproxima con un mínimo error la solución real del problema y por su facilidad es usado ampliamente. El método Predictor-Corrector de Adams de cuarto orden aproxima la solución dependiendo de los valores iniciales y de las iteraciones requeridas.

- El método de Runge-Kutta para sistemas de ecuaciones diferenciales de primer orden como procedimiento de solución del sistema de ecuaciones aplica el método de un paso (el método Runge Kutta de cuarto orden) a cada una de las ecuaciones antes de continuar con el siguiente paso.

- Una ecuación diferencial de orden superior puede resolverse con el método de Runge-Kutta para sistemas de ecuaciones diferenciales previa transformación de la ecuación diferencial de orden superior en un sistema de ecuaciones diferenciales de primer orden.

BIBLIOGRAFÍA

- Burden Richard L. y Faires J. Douglas. Análisis Numéricos. Séptima Edición. Editorial Thompson Learning. México D. F. 2002.

- Chapra Steven C. y Canales Raymond P. Métodos Numéricos para ingenieros. Editorial McGraw-Hill. México. 1998.

- Dennis G. Zill. Ecuaciones diferenciales con aplicaciones. Tercera Edición. Grupo Editorial Iberoamérica S.A. de C.V.1997.

- Dennis G. Zill. Ecuaciones diferenciales con aplicaciones de modelado. Novena Edición. CENGACE Learning Editores S.A. de C.V. 2009.

- Gutiérrez Robles, Juan Alberto; Olmos Gómez, Miguel Ángel y Casillas Gonzales, Juan Martin. Análisis numérico. Primera edición. McGraw-Hill/Interamericana editores, S.A. de C.V., México D.F., 2010.

- Mathews John H. y Fink Kurtis D. Métodos Numéricos con MATLAB. Tercera Edición. Editorial Prentice Hall. Madrid 2000.

- William Anthony Granville. Calculo diferencial e integral. Editorial Limusa, S. A. de C. V. Balderas 95, Primer piso, 06040, México, D. F. 1990.

- Cohen A.M. Análisis Numérico. Editorial Reverté, S.A., 1977.

- Ralston Anthony. Introducción al Análisis Numérico. México: Editorial Limusa, 1978.

ANEXOS

ANEXO Nº 1
PROGRAMA DEL MÉTODO DE NEWTON-RAPHSON

```
%Programa para obtener la solución de la ecuación f(x)=0
%dada la función diferenciable f y una aproximación inicial p0

function NewtonRaphson(p0,tol,m)

%Datos
%p0 es la aproximación inicial
%tol es la tolerancia o precisión deseada
%m es el número máximo de iteraciones
%Resultado
%p solución aproximada

p0=input('introduzca la aproximación inicial ')
tol=input('introduzca la tolerancia ')
m=input('introduzca el número máximo de iteraciones ')

i=1;
fprintf ('\n     i      pi     /pi-pi-1/');
while i<=m
   fp0=exp(p0)-3*p0^2;  %función a evaluar
   dfp0=exp(p0)-6*p0;   % derivada de la función

   p=p0-fp0/dfp0;
   t=abs (p-p0);
fprintf ('\n%d  %f   %f', i,p,t);
if t<tol
fprintf ('\n la solución aproximada es %g', p);

return;
end
   i=i+1;
   p0=p;
end
fprintf ('\n el método fracaso después de %d iteraciones, m');
```

SALIDA QUE CORRESPONDE AL EJEMPLO 1

```
NewtonRaphson

introduzca la aproximación inicial 0.5
p0 =
    0.5000

introduzca la tolerancia 0.001
tol =
     0.0010
introduzca el número máximo de iteraciones 4
m =
    4

i      pi     /pi-pi-1/
1  1.165089   0.665089
2  0.936227   0.228863
3  0.910397   0.025830
4  0.910008   0.000389

la solución aproximada es 0.910008
```

ANEXO Nº 2

PROGRAMA DEL MÉTODO DE ELIMINACIÓN GAUSSIANA CON SUSTITUCIÓN HACIA ATRÁS

```
%Programa para resolver un sistema de n ecuaciones lineales con
%n incógnitas mediante el método de eliminación gaussiana y
%sustitución hacia atrás

function EGSA(n,A)

%Datos
%n numero de incógnitas y de ecuaciones
%A matriz aumentada de los coeficientes
%Resultados
%x vector (x1,x2,...xn) de la solución del sistema

n=input('\n introduzca el número de incógnitas y ecuaciones n:')
A=input('\n introduzca la matriz aumentada de los coef. A:')

for i=1:n-1
if A(i,i)==0
      p=i+1;
    while (A(p,i)==0 & p~=n)
         p=p+1;
end
if p==n
   fprintf('\n no existe solución única');
   return;
end
else
      p=i;
end

if p~=i
    for k=1:n+1
         t=A(i,k);
         A(i,k)=A(p,k);
         A(p,k)=t;
    end
end
 for j=i+1:n
      m(j,i)=A(j,i)/A(i,i);
  for k=1:n+1
      A(j,k)=A(j,k)-m(j,i)*A(i,k);
  end
 end
end

if A(n,n)==0
   fprintf('\n no existe solución única')
   return;
end
```

```
    x(n)=A(n,n+1)/A(n,n);
for i=n-1:-1:1
    suma=0;
  for j=i+1:n
      suma=suma+A(i,j)*x(j);
  end
      x(i)=(A(i,n+1)-suma)/A(i,i);
end
    fprintf('\n la solución del sistema es ');
for i=1:n
 fprintf('\n x(%d)=%f',i,x(i));
end
```

SALIDA QUE CORRESPONDE AL EJEMPLO 2

```
EGSA

introduzca el número de incógnitas y ecuaciones n:4
n =
    4

introduzca la matriz aumentada de los coef. A:[1 2 -1 4 1;0 4 -2
0.25 4;1 2 6 3 3;0.5 1 -0.5 1 2]


           1.0000    2.0000   -1.0000    4.0000    1.0000
                0    4.0000   -2.0000    0.2500    4.0000
           1.0000    2.0000    6.0000    3.0000    3.0000
           0.5000    1.0000   -0.5000    1.0000    2.0000


la solución del sistema es

 x(1)=4.812500
 x(2)=1.129464
 x(3)=0.071429
 x(4)=-1.500000
```

ANEXO Nº 3
PROGRAMA DEL MÉTODO ITERATIVO DE GAUSS-SEIDEL

```
%Programa para resolver sistemas de n ecuaciones lineales con n
%incógnitas usando la técnica de iteración de gauss-seidel

function GaussSeidel(n,A,b,x0,tol,m)

%Datos
%n el numero de ecuaciones e incógnitas
%A la matriz de coeficientes
%b vector de términos independientes
%x0 vector de aproximación inicial
%tol la tolerancia
%m el número máximo de iteraciones
%Resultados
%la solución aproximada x1,…,xn o el mensaje de que rebaso el
numero de iteraciones

n=input('Introduzca el número de ec. e incógnitas n:   ');
A=input('Introduzca la matriz de coeficientes A:   ');
b=input('Introduzca el vector de términos independ. b:   ');
x0=input('Introduzca el vector de aprox. inicial x0:   ');
tol=input('Introduzca la tolerancia tol:   ');
m=input('Introduzca el número máximo de iteraciones m:   ');

k=1;
fprintf(' k  ');
for i=1:n
fprintf('   x%i(k)    ',i);
end
fprintf(' //x(k)-x(k-1)// \n');

while k<=m
for i=1:n
    suma1=0;
 for j=1:i-1

    suma1=suma1+A(i,j)*x(j);
 end
    suma2=0;
 for j=i+1:n

     suma2=suma2+A(i,j)*x0(j);
 end

 x(i)=(-suma1-suma2+ b(i))/A(i,i);
end

z=norm(x-x0,inf);
fprintf(' %i ',k);
 for i=1:n
 fprintf('  %f ',x(i));
 end
```

```
fprintf('  %f  \n ',z);
if z<tol
fprintf('\n la solución del sistema es \n')

 for i=1:n
 fprintf('\n x(%d)=%f \n',i,x(i));
  end
return
end
 k=k+1;
 for i=1:n
 x0(i)=x(i);
 end
end
fprintf('\n número máximo de iteraciones excedido\n')
```

SALIDA QUE CORRESPONDE AL EJEMPLO 3

```
GaussSeidel

Introduzca el número de ec. e incógnitas n: 4

Introduzca la matriz de coeficientes A: [4 1 -1 1; 1 4 -1 -1; -1
-2 5 1; 1 -1 1 3]

Introduzca el vector de términos independ. b: [-2 -1 0 1]

Introduzca el vector de aprox. inicial x0: [0 0 0 0]

Introduzca la tolerancia tol: 0.001

Introduzca el número máximo de iteraciones m: 8
```

k	x1(k)	x2(k)	x3(k)	x4(k)	//x(k)-x(k-1)//
1	-0.500000	-0.125000	-0.125000	0.500000	0.500000
2	-0.625000	0.000000	-0.225000	0.616667	0.125000
3	-0.710417	0.025521	-0.260313	0.665417	0.085417
4	-0.737812	0.035729	-0.273500	0.682347	0.027396
5	-0.747894	0.039185	-0.278211	0.688430	0.010082
6	-0.751457	0.040419	-0.279894	0.690590	0.003563
7	-0.752726	0.040855	-0.280492	0.691358	0.001269
8	-0.753176	0.041010	-0.280705	0.691630	0.000451
9	-0.753336	0.041066	-0.280780	0.691727	0.000160
10	-0.753393	0.041085	-0.280807	0.691762	0.000057
11	-0.753414	0.041092	-0.280817	0.691774	0.000020
12	-0.753421	0.041095	-0.280820	0.691778	0.000007

```
la solucion del sistema es

x(1)=-0.753421
x(2)=0.041095
x(3)=-0.280820
x(4)=0.691778
```

ANEXO Nº 4
PROGRAMA DEL MÉTODO DE NEWTON

```
%Programa para encontrar la solución de un sistema no lineal de
%dos ecuaciones F(x)=0 dada una aproximación inicial x0 donde
%F(x)=(f1(x), f2(x))

function Newton(x0,tol,max)

%Datos
%x0 es la aproximación inicial
%tol la tolerancia pedida
%max el numero máximo de iteraciones
%Resultados
%x es la aproximación a la solución que se obtiene
x0=input('introduzca la aproximación inicial (transpuesto):')
tol=input('introduzca la tolerancia pedida tol:')
max=input('introduzca el numero máximo de iteraciones m:')

n=length(x0);
k=1;
fprintf('\n    k    x(1)    x(2)    error   \n');
while (k<=max)

  B1=4*x0(1)^2-20*x0(1)+0.25*x0(2)^2+8; %función f1 evaluada en
x0
  B2=0.5*x0(1)*x0(2)^2+2*x0(1)-5*x0(2)+8; %función f1 evaluada
en x0
  B=[B1 B2];
  C=-1*B';
  %matriz jacobiana evaluada en x0
  J=[8*x0(1)-20 0.5*x0(2);0.5*x0(2)^2+2 x0(1)*x0(2)-5];
  Y=J\C;    %resuelve el sistema lineal J(x0)Y=-F(x0)
  x=x0+Y;
  err=norm(x-x0);
  fprintf('%d',k);
  fprintf('%10f',x(1));
  fprintf('%10f',x(2));
  fprintf('%10f',err);
  fprintf('\n');
  if (err<=tol)
    fprintf('\n la solucion del sistema es \n')
     for i=1:n
      fprintf('\n x(%d)=%f \n',i,x(i));
     end
     return;
  end
   k=k+1;
   x0=x;
end
 fprintf('\n número máximo de iteraciones excedido \n' );
```

SALIDA QUE CORRESPONDE AL EJEMPLO 4

```
Newton
introduzca la aproximación inicial (transpuesto):[0 0]'
x0 =
     0
     0

introduzca la tolerancia pedida tol:0.0001
tol =
     1.0000e-004

introduzca el numero máximo de iteraciones m:10
max =
       10

k    x(1)    x(2)    error
1  0.400000  1.760000  1.804882
2  0.495894  1.983423  0.243133
3  0.499988  1.999937  0.017013
4  0.500000  2.000000  0.000064

la solucion del sistema es

x(1)=0.500000
x(2)=2.000000
```

ANEXO Nº 5

PROGRAMA DEL MÉTODO DE RUNGE-KUTTA DE CUARTO ORDEN

```
%Programa para aproximar la solución del problema de valor
%inicial y'=f(t,y)con y(a)=d en M pasos en [a,b]mediante el
%método de Runge-Kutta de orden 4

function RungeKutta4(a,b,d,M)

%Datos
%a es el extremo izquierdo del intervalo
%b es el extremo derecho del intervalo
%d es la condición inicial y(a)
%M es el numero de pasos
%Resultado
%aproximación w a y en los (N+1) valores de t

a=input('\n introduzca el extremo izquierdo del intervalo a:')
b=input('\n introduzca el extremo derecho del intervalo b:')
d=input('\n introduzca la condición inicial y(a):')
M=input('\n introduzca el numero de pasos M:')
h=(b-a)/M;
t=a;
w=d;
fprintf('\n    t     w');
fprintf('\n %6.1f  %f ',t,w);
for i=1:M
   p1=feval('fun',t,w);
   k1=h*p1;

   p2=feval('fun',t+h/2,w+k1/2);
   k2=h*p2;

   p3=feval('fun',t+h/2,w+k2/2);
   k3=h*p3;

   p4=feval('fun',t+h,w+k3);
   k4=h*p4;

   w=w+(k1+2*k2+2*k3+k4)/6;
   t=a+i*h;
   fprintf('\n %6.1f    %f ',t,  w);
end


%la función de trabajo se archiva aparte y con distinto nombre

function z=fun(t,w)
z=1+(1/t)*w; % función a evaluar
```

SALIDA QUE CORRESPONDE AL EJEMPLO 5

```
RungeKutta4

introduzca el extremo izquierdo del intervalo a:1
a =
    1

introduzca el extremo derecho del intervalo b:2
b =
    2

introduzca la condición inicial y(a):2
d =
    2

introduzca el numero de pasos M:4
M =
    4

 t         w
1.0    2.000000
1.3    2.778909
1.5    3.608165
1.8    4.479285
2.0    5.386243
```

ANEXO Nº 6

EL PROGRAMA DEL METODO PREDICTOR-CORRECTOR DE ADAMS DE CUARTO ORDEN

```
%Programa para aproximar la solución del problema de valor
%inicial y'=f(t,y)con y(a)=d en (N+1) números uniformemente
%espaciados en el intervalo [a,b] mediante el Método Predictor-
%Corrector de cuarto orden de Adams

function PCAdams4(a,b,N,d)

%Datos
%a es el extremo izquierdo del intervalo
%b es el extremo derecho del intervalo
%N es el entero
%d es la condición inicial y(a)=d
%Resultado
%aproximación w a y en los (N+1) valores de t
a=input('\n introduzca el extremo izquierdo del intervalo a:')
b=input('\n introduzca el extremo derecho del intervalo b:')
N=input('\n introduzca el entero N:')
d=input('\n introduzca la condición inicial: y(a)')

h=(b-a)/N;
t(1)=a;
w(1)=d;
fprintf('\n    ti    wi');
fprintf('\n %6.1f  %f ',t(1),w(1));
for i=2:4
   p1=feval('func',t(i-1),w(i-1));
   k1=h*p1;
   p2=feval('func',t(i-1)+h/2,w(i-1)+k1/2);
   k2=h*p2;

   p3=feval('func',t(i-1)+h/2,w(i-1)+k2/2);
   k3=h*p3;

   p4=feval('func',t(i-1)+h,w(i-1)+k3);
   k4=h*p4;

   w(i)=w(i-1)+(k1+2*k2+2*k3+k4)/6;
   t(i)=a+(i-1)*h;
   fprintf('\n %6.1f    %f ',t(i),w(i));
end

for i=5:N+1
   r=a+(i-1)*h;
   q4=feval('func',t(4),w(4));
   q3=feval('func',t(3),w(3));
   q2=feval('func',t(2),w(2));
   q1=feval('func',t(1),w(1));
```

```
    s=w(4)+h*(55*q4-59*q3+37*q2-9*q1)/24;
    q=feval('func',r,s);
    s=w(4)+h*(9*q+19*q4-5*q3+q2)/24;
    fprintf('\n %6.1f      %f ',r,s);
    for j=1:3
       t(j)=t(j+1);
       w(j)=w(j+1);
    end

       t(4)=r;
       w(4)=s;
end


%la función de trabajo se archiva aparte y con distinto nombre

function z=func(t,w)

z=-5*w+5*t*t+2*t;  %función a evaluar
```

SALIDA QUE CORRESPONDE AL EJEMPLO 6

```
PCAdams4

introduzca el extremo izquierdo del intervalo a:0
a =
     0

introduzca el extremo derecho del intervalo b:1
b =
     1

introduzca el entero N:5
N =
     5

introduzca la condición inicial: y(a)0.5
d =
     0.5000

      ti       wi
     0.0     0.500000
     0.2     0.228333
     0.4     0.231458
     0.6     0.387630
     0.8     0.638872
     1.0     0.991353
```

ANEXO Nº 7

PROGRAMA DEL MÉTODO DE RUNGE-KUTTA PARA SISTEMAS DE ECUACIONES DIFERENCIALES DE PRIMER ORDEN

```
%programa para aproximar la solución de un sistema de orden 3 de
%los problemas de valor inicial de primer orden
%u'j=fj(t,u1,u2,u3) a<=t<=b, con uj(a)=dj, para j=1,2,3 en (m+1)
%números uniformemente espaciados en el intervalo [a,b]
%mediante el método de Runge-Kutta para sistemas de ecuaciones
%diferenciables

function RungeKuttaSEDc(a,b,m,d)

%datos
%a es el extremo izquierdo del intervalo
%b es el extremo derecho del intervalo
%m es el número de subintervalos
%d vector de condiciones iniciales d1,d2,d3
%resultado
%aproximación wj a uj(t) en los (m+1) valores de t.

a=input('\n introduzca el extremo izquierdo del intervalo a:')
b=input('\n introduzca el extremo derecho del intervalo b:')
m=input('\n introduzca el numero de subintervalos m:')
d=input('\n introduzca el vector de  cond. inic. d1,...,dn:')

n=3;
h=(b-a)/m;
t=a;
for j=1:n
   w(j)=d(j);
end
fprintf('\n    ti       w1i       w2i       w3i  ');
fprintf('\n %6.1f  %f   %f   %f ',t,w(1),w(2),w(3) );
for i=1:m

     p(1)=feval('func1',t,w(1),w(2),w(3));
     p(2)=feval('func2',t,w(1),w(2),w(3));
     p(3)=feval('func3',t,w(1),w(2),w(3));
      for j=1:n
          k(1,j)=h*p(j);
      end


p(1)=feval('func1',t+h/2,w(1)+1/2*k(1,1),w(2)+1/2*k(1,2),w(3)+1/
2*k(1,3));

p(2)=feval('func2',t+h/2,w(1)+1/2*k(1,1),w(2)+1/2*k(1,2),w(3)+1/
2*k(1,3));

p(3)=feval('func3',t+h/2,w(1)+1/2*k(1,1),w(2)+1/2*k(1,2),w(3)+1/
2*k(1,3));
```

```
      for j=1:n
          k(2,j)=h*p(j);
      end


p(1)=feval('func1',t+h/2,w(1)+1/2*k(2,1),w(2)+1/2*k(2,2),w(3)+1/
2*k(2,3));

p(2)=feval('func2',t+h/2,w(1)+1/2*k(2,1),w(2)+1/2*k(2,2),w(3)+1/
2*k(2,3));

p(3)=feval('func3',t+h/2,w(1)+1/2*k(2,1),w(2)+1/2*k(2,2),w(3)+1/
2*k(2,3));
      for j=1:n
         k(3,j)=h*p(j);
      end


p(1)=feval('func1',t+h,w(1)+k(3,1),w(2)+k(3,2),w(3)+k(3,3));

p(2)=feval('func2',t+h,w(1)+k(3,1),w(2)+k(3,2),w(3)+k(3,3));

p(3)=feval('func3',t+h,w(1)+k(3,1),w(2)+k(3,2),w(3)+k(3,3));
      for j=1:n
         k(4,j)=h*p(j);
      end
     for j=1:n
         w(j)=w(j)+(k(1,j)+2*k(2,j)+2*k(3,j)+k(4,j))/6;
     end
         t=a+i*h;
         fprintf('\n %6.1f   %f   %f    %f ',t,w(1),w(2),w(3) );
 end



%Cada función se archiva aparte y con distinto nombre

function z=fun1(t,y1,y2,y3) %primera función a evaluar
z=y2;

function z=fun2(t,y1,y2,y3)  %segunda función a evaluar
z=-y1-2*exp(t)+1;

function z=fun3(t,y1,y2,y3)  %tercera función a evaluar
z=-y1-exp(t)+1;
```

SALIDA QUE CORRESPONDE AL EJEMPLO 7

```
RungeKuttaSEDc

introduzca el extremo izquierdo del intervalo a:0

a =
     0

introduzca el extremo derecho del intervalo b:2

b =
     2

introduzca el numero de subintervalos m:4

m =
     4

introduzca el vector de  cond. inic. d1,...,dn:[1 0 1]

d =

    1     0     1



     ti       w1i         w2i          w3i
    0.0    1.000000   0.000000    1.000000
    0.5    0.707871  -1.249887    0.398849
    1.0   -0.336918  -3.017642   -0.299323
    1.5   -2.413327  -5.405233   -0.923469
    2.0   -5.894790  -8.709705   -1.320512
```

ANEXO 8

PROGRAMA DEL MÉTODO DE RUNGE-KUTTA PARA SISTEMAS DE ECUACIONES DIFERENCIALES DE PRIMER ORDEN

```
%programa para aproximar la solución de un sistema de orden 3 de
%los problemas de valor inicial de primer orden
%u'j=fj(t,u1,u2,u3) a<=t<=b, con uj(a)=dj, para j=1,2,3 en (m+1)
%números uniformemente espaciados en el intervalo [a,b]
%mediante el método de Runge-Kutta para sistemas de ecuaciones
%diferenciables

function RungeKuttaSEDcc(a,b,m,d)

%datos
%a es el extremo izquierdo del intervalo
%b es el extremo derecho del intervalo
%m es el número de subintervalos
%d vector de condiciones iniciales d1,d2,d3
%resultado
%aproximación wj a uj(t) en los (m+1) valores de t.

a=input('\n introduzca el extremo izquierdo del intervalo a:')
b=input('\n introduzca el extremo derecho del intervalo b:')
m=input('\n introduzca el numero de subintervalos m:')
d=input('\n introduzca el vector de  cond. inic. d1,...,dn:')

n=3;
h=(b-a)/m;
t=a;
for j=1:n
   w(j)=d(j);
end
fprintf('\n    ti       w1i       w2i       w3i  ');
fprintf('\n %6.1f  %f   %f   %f ',t,w(1),w(2),w(3) );
for i=1:m

     p(1)=feval('funcc1',t,w(1),w(2),w(3));
     p(2)=feval('funcc2',t,w(1),w(2),w(3));
     p(3)=feval('funcc3',t,w(1),w(2),w(3));
      for j=1:n
          k(1,j)=h*p(j);
      end


p(1)=feval('funcc1',t+h/2,w(1)+1/2*k(1,1),w(2)+1/2*k(1,2),w(3)+1
/2*k(1,3));

p(2)=feval('funcc2',t+h/2,w(1)+1/2*k(1,1),w(2)+1/2*k(1,2),w(3)+1
/2*k(1,3));

p(3)=feval('funcc3',t+h/2,w(1)+1/2*k(1,1),w(2)+1/2*k(1,2),w(3)+1
/2*k(1,3));
```

```
      for j=1:n
          k(2,j)=h*p(j);
      end


p(1)=feval('funcc1',t+h/2,w(1)+1/2*k(2,1),w(2)+1/2*k(2,2),w(3)+1
/2*k(2,3));

p(2)=feval('funcc2',t+h/2,w(1)+1/2*k(2,1),w(2)+1/2*k(2,2),w(3)+1
/2*k(2,3));

p(3)=feval('funcc3',t+h/2,w(1)+1/2*k(2,1),w(2)+1/2*k(2,2),w(3)+1
/2*k(2,3));
      for j=1:n
         k(3,j)=h*p(j);
      end


p(1)=feval('funcc1',t+h,w(1)+k(3,1),w(2)+k(3,2),w(3)+k(3,3));

p(2)=feval('funcc2',t+h,w(1)+k(3,1),w(2)+k(3,2),w(3)+k(3,3));

p(3)=feval('funcc3',t+h,w(1)+k(3,1),w(2)+k(3,2),w(3)+k(3,3));
      for j=1:n
         k(4,j)=h*p(j);
      end
     for j=1:n
         w(j)=w(j)+(k(1,j)+2*k(2,j)+2*k(3,j)+k(4,j))/6;
     end
         t=a+i*h;
         fprintf('\n %6.1f   %f   %f    %f ',t,w(1),w(2),w(3) );
 end



% Cada función se archiva aparte y con distinto nombre

function z=funcc1(t,y1,y2,y3) %primera función a evaluar
z=y2;

function z=funcc2(t,y1,y2,y3)  %segunda función a evaluar
z=y3;

function z=funcc3(t,y1,y2,y3)  %tercera función a evaluar
z=-2*y3+y2+2*y1+exp(t);
```

SALIDA QUE CORRESPONDE AL EJEMPLO 8

```
RungeKuttaSEDcc

introduzca el extremo izquierdo del intervalo a:0

a =
     0

introduzca el extremo derecho del intervalo b:3

b =
     3

introduzca el numero de subintervalos m:15

m =
    15

introduzca el vector de  cond. inic. d1,...,dn:[1 2 0]

d =
     1     2     0

      ti      w1i         w2i         w3i
     0.0   1.000000   2.000000    0.000000
     0.2   1.406337   2.094395    0.919616
     0.4   1.849181   2.361889    1.747229
     0.6   2.361909   2.792901    2.567577
     0.8   2.977556   3.393107    3.450121
     1.0   3.731627   4.181249    4.457219
     1.2   4.664604   5.188356    5.650332
     1.4   5.824428   6.458085    7.095094
     1.    7.269132   8.048019    8.865841
     1.8   9.069833  10.031843   11.050033
     2.0  11.314246  12.502434   13.752964
     2.2  14.110911  15.575943   17.103040
     2.4  17.594350  19.397023   21.257977
     2.6  21.931402  24.145402   26.412221
     2.8  27.329028  30.044109   32.805972
     3.0  34.043957  37.369687   40.736233
```

ANEXO 9
TABLAS DE VENTAJAS Y DESVENTAJAS DE LOS METODOS NUMERICOS ESTUDIADOS PARA LA SOLUCION DE PROBLEMAS MODELADOS MATEMATICAMENTE

1. PARA UNA ECUACION NO LINEAL

METODO NUMERICO	VALORES INICIALES	CONVERGENCIA	ESFUERZO COMPUTACIONAL	COMENTARIOS
NEWTON-RAPHSON	1	RAPIDA	FACIL	NECESITA EVALUACION DE $f'(x)$

2. PARA UN SISTEMA DE ECUACIONES LINEALES

METODOS NUMERICOS	ESTABILIDAD	PRECISION	RANGO DE APLICACIÓN	COMPLEJIDAD DE PROGRAMACION
ELIMINACION GAUSSIANA CON SUSTITUCION HACIA ATRAS	–	AFECTADA POR ERRORES DE REDONDEO	GENERAL	MODERADA
GAUSS-SEIDEL	PUEDE NO CONVERGER SI NO ES DIAGONALMENTE DOMINANTE	EXCELENTE	APROPIADA SOLO PARA SISTEMAS DIAGONALMENTE DOMINANTE	FACIL

3. PARA UN SISTEMA DE ECUACIONES NO LINEALES

METODO NUMERICO	VALORES NUMERICOS	CONVERGENCIA	ESFUERZO COMPUTACIONAL	COMENTARIOS
NEWTON	DEPENDE DE CUANTAS INCOGNITAS TENGA EL SISTEMA	MUY BUENA	MODERADO	NECESITA EVALUAR LA INVERSA DE LA MATRIZ JACOBIANA

4. PARA UNA ECUACION DIFERENCIAL ORDINARIA DE PRIMER ORDEN CON VALOR INICIAL

METODOS NUERICOS	VALORES INICIALES	ITERACIONES REQUERIDAS	CAMBIO DE TAMAÑO DE PASO	DIFICULTAD DE PROGRAMACION	COMENTARIOS
EL METODO DE UN PASO DE RUNGE-KUTTA DE CUARTO ORDEN	1	NO	FACIL	MODERADA	AMPLIAMENTE USADO
EL METODO DE MULTIPLES PASOS DE ADAMS DE CUARTO ORDEN	4	SI	DIFICIL	MODERADA A EXTENSA CON TAMAÑO DE PASO VARIABLE	–

5. PARA UN SISTEMAS DE ECUACIONES DIFERENCIALES DE PRIMER ORDEN

METODOS NUERICOS	VALORES INICIALES	ITERACIONES REQUERIDAS	CAMBIO DE TAMAÑO DE PASO	DIFICULTAD DE PROGRAMACION	COMENTARIOS
EL METODO DE RUNGE-KUTTA PARA SISTEMAS DE ECUACIONES DIFERENCIALES	1	NO	FACIL	MODERADA	AMPLIAMENTE USADO

ANEXO 10

PROGRAMA DEL MÉTODO DE NEWTON-RAPHSON PARA EL CASO 1

```
%Programa para obtener la solución de la ecuación f(x)=0
%dada la función diferenciable f y una aproximación inicial p0

function NewtonRaphsonI(p0,tol,m)

%Datos
%p0 es la aproximación inicial
%tol es la tolerancia o precisión deseada
%m es el número máximo de iteraciones
%Resultado
%p solución aproximada

p0=input('introduzca la aproximación inicial ')
tol=input('introduzca la tolerancia ')
m=input('introduzca el número máximo de iteraciones ')

i=1;
fprintf ('\n      i      pi     /pi-pi-1/');
while i<=m
   fp0=-135000+(1000/p0)*(1-(1+p0)^(-360)); %función a evaluar
   dfp0=(1000/p0)*((-1/p0)*(1-(1+p0)^(-360))+360*(1+p0)^(-361));
% derivada de la función
   p=p0-fp0/dfp0;
   t=abs (p-p0);
fprintf ('\n%d  %f   %f', i,p,t);
if t<tol
fprintf ('\n la solución aproximada es %f', p);

return;
end
   i=i+1;
   p0=p;
end
fprintf ('\n el método fracaso después de %d iteraciones',m);
```

SALIDA QUE CORRESPONDE AL CASO 1

```
NewtonRaphsonI

introduzca la aproximación inicial 0.01

p0 =
    0.0100

introduzca la tolerancia 0.0001

tol =
     1.0000e-004

introduzca el número máximo de iteraciones 4

m =
    4

i      pi     /pi-pi-1/
1  0.005672   0.004328
2  0.006646   0.000973
3  0.006749   0.000103
4  0.006750   0.000001

la solución aproximada es 0.006750
```

ANEXO 11

PROGRAMA DEL MÉTODO DE ELIMINACIÓN GAUSSIANA CON SUSTITUCION HACIA ATRÁS PARA EL CASO 2

```
%Programa para resolver un sistema de n ecuaciones lineales con
%n incógnitas mediante el método de eliminación gaussiana y
%sustitución hacia atrás

function EGSA(n,A)

%Datos
%n numero de incógnitas y de ecuaciones
%A matriz aumentada de los coeficientes
%Resultados
%x vector (x1,x2,...xn) de la solución del sistema

n=input('\n introduzca el número de incógnitas y ecuaciones n:')
A=input('\n introduzca la matriz aumentada de los coef. A:')

for i=1:n-1
if A(i,i)==0
      p=i+1;
    while (A(p,i)==0 & p~=n)
         p=p+1;
end
if p==n
   fprintf('\n no existe solución única');
   return;
end
else
      p=i;
end

if p~=i
    for k=1:n+1
         t=A(i,k);
         A(i,k)=A(p,k);
         A(p,k)=t;
    end
end
 for j=i+1:n
      m(j,i)=A(j,i)/A(i,i);
  for k=1:n+1
      A(j,k)=A(j,k)-m(j,i)*A(i,k);
  end
 end
end

if A(n,n)==0
   fprintf('\n no existe solución única')
   return;
end

    x(n)=A(n,n+1)/A(n,n);
```

```
for i=n-1:-1:1
    suma=0;
  for j=i+1:n
      suma=suma+A(i,j)*x(j);
  end
      x(i)=(A(i,n+1)-suma)/A(i,i);
end
    fprintf('\n la solución del sistema es ');
for i=1:n
 fprintf('\n x(%d)=%f',i,x(i));
end
```

SALIDA QUE CORRESPONDE AL CASO 2

```
EGSA

introduzca el número de incógnitas y ecuaciones n:3

n =
    3

introduzca la matriz aumentada de los coef. A:[0.52 0.20
0.25 4800;0.30 0.50 0.20 5800;0.18 0.30 0.55 5700]

A =
    1.0e+003 *


0.0005    0.0002    0.0003    4.8000
0.0003    0.0005    0.0002    5.8000
0.0002    0.0003    0.0006    5.7000

la solución del sistema es

 x(1)=4005.813953
 x(2)=7131.395349
 x(3)=5162.790698
```

ANEXO 12

PROGRAMA DEL MÉTODO DE GAUSS-SEIDEL PARA EL CASO 2

```
%Programa para resolver sistemas de n ecuaciones lineales con n
%incógnitas usando la técnica de iteración de gauss-seidel

function GaussSeidel(n,A,b,x0,tol,m)

%Datos
%n el numero de ecuaciones e incógnitas
%A la matriz de coeficientes
%b vector de términos independientes
%x0 vector de aproximación inicial
%tol la tolerancia
%m el número máximo de iteraciones
%Resultados
%la solución aproximada x1,…,xn o el mensaje de que rebaso el
numero de iteraciones

n=input('Introduzca el número de ec. e incógnitas n:   ');
A=input('Introduzca la matriz de coeficientes A:   ');
b=input('Introduzca el vector de términos independ. b:   ');
x0=input('Introduzca el vector de aprox. inicial x0:   ');
tol=input('Introduzca la tolerancia tol:   ');
m=input('Introduzca el número máximo de iteraciones m:   ');

k=1;
fprintf(' k  ');
for i=1:n
fprintf('   x%i(k)     ',i);
end
fprintf(' //x(k)-x(k-1)// \n');

while k<=m
for i=1:n
    suma1=0;
 for j=1:i-1

    suma1=suma1+A(i,j)*x(j);
 end
    suma2=0;
 for j=i+1:n

     suma2=suma2+A(i,j)*x0(j);
 end

 x(i)=(-suma1-suma2+ b(i))/A(i,i);
end

z=norm(x-x0,inf);
fprintf(' %i ',k);
 for i=1:n
 fprintf('  %f ',x(i));
 end
```

```
fprintf('  %f  \n ',z);
if z<tol
fprintf('\n la solución del sistema es \n')

 for i=1:n
 fprintf('\n x(%d)=%f \n',i,x(i));
  end
return
end
 k=k+1;
 for i=1:n
 x0(i)=x(i);
 end
end
fprintf('\n número máximo de iteraciones excedido\n')
```

SALIDA QUE CORRESPONDE AL CASO 2

```
GaussSeidel

Introduzca el número de ec. e incógnitas n:   3

Introduzca la matriz de coeficientes A:   [0.52 0.20 0.25;0.30
0.50 0.20 ;0.18 0.30 0.55]

Introduzca el vector de términos independ. b:   [4800 5800 5700]

Introduzca el vector de aprox. inicial x0:   [0 0 0]

Introduzca la tolerancia tol:   0.0001

Introduzca el número máximo de iteraciones m:   14

  k        x1(k)          x2(k)          x3(k)       //x(k)-x(k-1)//
  1    9230.769231    6061.538462    4036.363636    9230.769231
  2    4958.848843    7010.145239    4917.024793    4271.920387
  3    4170.605296    7130.826905    5109.169046     788.243548
  4    4031.812226    7137.245046    5151.091428     138.793070
  5    4009.188719    7134.050197    5160.238130      22.623507
  6    4006.020054    7132.292716    5162.233774       3.168665
  7    4005.736564    7131.664552    5162.669187       0.628164
  8    4005.768832    7131.471026    5162.764186       0.193526
  9    4005.797593    7131.415770    5162.784913       0.055256
  10   4005.808880    7131.400707    5162.789436       0.015063
  11   4005.812500    7131.396726    5162.790422       0.003981
  12   4005.813556    7131.395697    5162.790638       0.001057
  13   4005.813848    7131.395436    5162.790685       0.000292
  14   4005.813926    7131.395370    5162.790695       0.000078

 la solución del sistema es
 x(1)=4005.813926
 x(2)=7131.395370
 x(3)=5162.790695
```

ANEXO 13
PROGRAMA DEL MÉTODO DE NEWTON PARA EL CASO 3

```
%Programa para encontrar la solución de un sistema no lineal de
%dos ecuaciones F(x)=0 dada una aproximación inicial x0 donde
%F(x)=(f1(x), f2(x))

function Newton(x0,tol,max)

%Datos
%x0 es la aproximación inicial
%tol la tolerancia pedida
%max el numero máximo de iteraciones
%Resultados
%x es la aproximación a la solución que se obtiene
x0=input('introduzca las aproximaciones iniciales
(transpuesto):')
tol=input('introduzca la tolerancia pedida tol:')
max=input('introduzca el numero máximo de iteraciones m:')

n=length(x0);
k=1;
fprintf('\n    k    x1    x2    x3    error   \n');
while (k<=max)

  B1=75-x0(1)-2.35*exp(-3)*(x0(2)+x0(3))^(1.75); %función f1
evaluada en x0
  B2= x0(1)-20-4.67*exp(-3)*x0(2)^(1.75); %función f2 evaluada
en x0
  B3=x0(1)-15-3.72*exp(-2)*x0(3)^(1.75); %función f3 evaluada en
x0
  B=[B1 B2 B3];
  C=-1*B';
  %matriz jacobiana evaluada en x0
  J=[-1 -0.204749318*(x0(2)+x0(3))^(0.75) -
0.204749318*(x0(2)+x0(3))^(0.75); 1 -0.406884816*x0(2)^(0.75) 0;
1 0 -0.881032693*x0(3)^(0.75)];
  Y=J\C;    %resuelve el sistema lineal J(x0)Y=-F(x0)
  x=x0+Y;
  error=norm(x-x0);
  fprintf('%d',k);
  fprintf('%10f',x(1));
  fprintf('%10f',x(2));
  fprintf('%10f',x(3));
  fprintf('%10f',error);
  fprintf('\n');
  if (error<=tol)
    fprintf('\n la solucion del sistema es \n')
     for i=1:n
      fprintf('\n x(%d)=%f \n',i,x(i));
     end
     return;
  end
   k=k+1;
```

```
   x0=x;
end
 fprintf('\n número máximo de iteraciones excedido \n' );
```

SALIDA QUE CORRESPONDE AL CASO 3

```
NewtonI

introduzca las aproximaciones iniciales (transpuesto):[50 10 7]'

x0 =

     50
     10
      7

introduzca la tolerancia pedida tol:0.0001

tol =
     1.0000e-004


introduzca el numero máximo de iteraciones m:10

max =
      10


k    x1          x2         x3         error
1 43.958408 14.756674 10.637639  8.506421
2 43.959574 14.145360 10.139399  0.788637
3 43.959652 14.135549 10.130307  0.013376
4 43.959652 14.135547 10.130304  0.000004

la solucion del sistema es

 x(1)=43.959652
x(2)=14.135547
x(3)=10.130304
```

ANEXO 14

PROGRAMA DEL MÉTODO DE RUNGE-KUTTA DE CUARTO ORDEN PARA EL CASO 1

```
%Programa para aproximar la solución del problema de valor
%inicial y'=f(t,y)con y(a)=d en M pasos en [a,b]mediante el
%método de Runge-Kutta de orden 4

function RungeKutta41(a,b,d,M)

%Datos
%a es el extremo izquierdo del intervalo
%b es el extremo derecho del intervalo
%d es la condición inicial y(a)
%M es el numero de pasos
%Resultado
%aproximación w a y en los (N+1) valores de t

a=input('\n introduzca el extremo izquierdo del intervalo a:')
b=input('\n introduzca el extremo derecho del intervalo b:')
d=input('\n introduzca la condición inicial y(a):')
M=input('\n introduzca el numero de pasos M:')
h=(b-a)/M;
t=a;
w=d;
fprintf('\n    t     w');
fprintf('\n %6.1f  %f ',t,w);
for i=1:M
   p1=feval('fun1',t,w);
   k1=h*p1;

   p2=feval('fun1',t+h/2,w+k1/2);
   k2=h*p2;

   p3=feval('fun1',t+h/2,w+k2/2);
   k3=h*p3;

   p4=feval('fun1',t+h,w+k3);
   k4=h*p4;

   w=w+(k1+2*k2+2*k3+k4)/6;
   t=a+i*h;
   fprintf('\n %6.1f    %f ',t,  w);
end


%La función se archiva aparte y con distinto nombre

function z=fun1(t,w)

z=0.00001*20000*w-0.00001*w*w; %función a evaluar
```

SALIDA QUE CORRESPONDE AL CASO 1

```
RungeKutta41

introduzca el extremo izquierdo del intervalo a:0

a =

    0

introduzca el extremo derecho del intervalo b:30

b =
    30

introduzca la condición inicial y(a):6000

d =
    6000

introduzca el numero de pasos M:60

M =

    60

 t         w           9.5  14825.868262      20.0 19180.300296
0.0   6000.000000      10.0 15200.081843      20.5 19255.400870
0.5   6428.207310      10.5 15555.344526      21.0 19323.863313
1.0   6871.974217      11.0 15891.420081      21.5 19386.231598
1.5   7329.830900      11.5 16208.278113      22.0 19443.012722
2.0   7800.068110      12.0 16506.072364      22.5 19494.677817
2.5   8280.756475      12.5 16785.117442      23.0 19541.663527
3.0   8769.773704      13.0 17045.864982      23.5 19584.373568
3.5   9264.839118      13.5 17288.880079      24.0 19623.180422
4.0   9763.554437      14.0 17514.818668      24.5 19658.427093
4.5  10263.449358      14.5 17724.406376      25.0 19690.428898
5.0  10762.030135      15.0 17918.419190      25.5 19719.475253
5.5  11256.829135      15.5 18097.666172      26.0 19745.831433
6.0  11745.453302      16.0 18262.974309      26.5 19769.740283
6.5  12225.629526      16.5 18415.175512      27.0 19791.423857
7.0  12695.245120      17.0 18555.095688      27.5 19811.085000
7.5  13152.381980      17.5 18683.545759      28.0 19828.908834
8.0  13595.343378      18.0 18801.314474      28.5 19845.064168
8.5  14022.672822      18.5 18909.162825      29.0 19859.704824
9.0  14433.164879      19.0 19007.819886      29.5 19872.970867
                       19.5 19097.979881      30.0 19884.989762
```

ANEXO 15

PROGRAMA DEL MÉTODO DE PREDICTOR-CORRECTOR DE ADAMS DE CUARTO ORDEN PARA EL CASO 1

```
%Programa para aproximar la solución del problema de valor
%inicial y'=f(t,y)con y(a)=d en (N+1) números uniformemente
%espaciados en el intervalo [a,b] mediante el Método Predictor-
%Corrector de cuarto orden de Adams

function PCAdams42(a,b,N,d)

%Datos
%a es el extremo izquierdo del intervalo
%b es el extremo derecho del intervalo
%N es el entero
%d es la condición inicial y(a)=d
%Resultado
%aproximación w a y en los (N+1) valores de t
a=input('\n introduzca el extremo izquierdo del intervalo a:')
b=input('\n introduzca el extremo derecho del intervalo b:')
N=input('\n introduzca el entero N:')
d=input('\n introduzca la condición inicial: y(a)')

h=(b-a)/N;
t(1)=a;
w(1)=d;
fprintf('\n    ti    wi');
fprintf('\n %6.1f  %f ',t(1),w(1));
for i=2:4
   p1=feval('fun2',t(i-1),w(i-1));
   k1=h*p1;
   p2=feval('fun2',t(i-1)+h/2,w(i-1)+k1/2);
   k2=h*p2;

   p3=feval('fun2',t(i-1)+h/2,w(i-1)+k2/2);
   k3=h*p3;

   p4=feval('fun2',t(i-1)+h,w(i-1)+k3);
   k4=h*p4;

   w(i)=w(i-1)+(k1+2*k2+2*k3+k4)/6;
   t(i)=a+(i-1)*h;
   fprintf('\n %6.1f    %f ',t(i),w(i));
end

for i=5:N+1
   r=a+(i-1)*h;
   q4=feval('fun2',t(4),w(4));
   q3=feval('fun2',t(3),w(3));
   q2=feval('fun2',t(2),w(2));
   q1=feval('fun2',t(1),w(1));

   s=w(4)+h*(55*q4-59*q3+37*q2-9*q1)/24;
   q=feval('fun2',r,s);
   s=w(4)+h*(9*q+19*q4-5*q3+q2)/24;
```

```
   fprintf('\n %6.1f     %f ',r,s);
   for j=1:3
      t(j)=t(j+1);
      w(j)=w(j+1);
   end

      t(4)=r;
      w(4)=s;
end

%La función se archiva aparte y con distinto nombre

function z=fun2(t,w)

z=0.00001*20000*w-0.00001*w*w; %función a evaluar
```

SALIDA QUE CORRESPONDE AL CASO 1

```
PCAdams42

introduzca el extremo izquierdo del intervalo a:0
a =
     0
introduzca el extremo derecho del intervalo b:30
b =
    30
introduzca el entero N:60
N =
     60
introduzca la condición inicial: y(a)6000
d =
     6000
 ti      wi              9.5  14825.879524      20.0 19180.299517
0.0   6000.000000       10.0 15200.092282      20.5 19255.400202
0.5   6428.207310       10.5 15555.353982      21.0 19323.862764
1.0   6871.974217       11.0 15891.428445      21.5 19386.231173
1.5   7329.830900       11.5 16208.285327      22.0 19443.012419
2.0   7800.068662       12.0 16506.078417      22.5 19494.677633
2.5   8280.757787       12.5 16785.122365      23.0 19541.663455
3.0   8769.775972       13.0 17045.868839      23.5 19584.373601
3.5   9264.842514       13.5 17288.882960      24.0 19623.180550
4.0   9763.559093       14.0 17514.820680      24.5 19658.427306
4.5   10263.455355      14.5 17724.407636      25.0 19690.429186
5.0   10762.037489      15.0 17918.419819      25.5 19719.475605
5.5   11256.837794      15.5 18097.666289      26.0 19745.831841
6.0   11745.463148      16.0 18262.974025      26.5 19769.740735
6.5   12225.640376      16.5 18415.174930      27.0 19791.424346
7.0   12695.256741      17.0 18555.094900      27.5 19811.085518
7.5   13152.394101      17.5 18683.544844      28.0 19828.909372
8.0   13595.355706      18.0 18801.313499      28.5 19845.064721
8.5   14022.685063      18.5 18909.161844      29.0 19859.705385
9.0   14433.176755      19.0 19007.818943      29.5 19872.971431
                        19.5 19097.979008      30.0 19884.990325
```

ANEXO 16

PROGRAMA DEL MÉTODO DE RUNGE-KUTTA PARA SISTEMAS DE ECUACIONES DIFERENCIALES PARA EL CASO 2

```
%programa para aproximar la solución de un sistema de orden 3 de
%los problemas de valor inicial de primer orden
%u'j=fj(t,u1,u2,u3) a<=t<=b, con uj(a)=dj, para j=1,2,3 en (m+1)
%números uniformemente espaciados en el intervalo [a,b]
%mediante el método de Runge-Kutta para sistemas de ecuaciones
%diferenciables

function RungeKuttaSEDccc(a,b,m,d)

%datos
%a es el extremo izquierdo del intervalo
%b es el extremo derecho del intervalo
%m es el número de subintervalos
%d vector de condiciones iniciales d1,d2,d3
%resultado
%aproximación wj a uj(t) en los (m+1) valores de t.

a=input('\n introduzca el extremo izquierdo del intervalo a:')
b=input('\n introduzca el extremo derecho del intervalo b:')
m=input('\n introduzca el numero de subintervalos m:')
d=input('\n introduzca el vector de  cond. inic. d1,d2,d3:')

n=3;
h=(b-a)/m;
t=a;
for j=1:n
   w(j)=d(j);
end
fprintf('\n    ti       Si         Ii          Ri  ');
fprintf('\n %6.1f   %f   %f    %f ',t,w(1),w(2),w(3) );
for i=1:m

     p(1)=feval('funccc1',t,w(1),w(2),w(3));
     p(2)=feval('funccc2',t,w(1),w(2),w(3));
     p(3)=feval('funccc3',t,w(1),w(2),w(3));
      for j=1:n
          k(1,j)=h*p(j);
      end


p(1)=feval('funccc1',t+h/2,w(1)+1/2*k(1,1),w(2)+1/2*k(1,2),w(3)+
1/2*k(1,3));

p(2)=feval('funccc2',t+h/2,w(1)+1/2*k(1,1),w(2)+1/2*k(1,2),w(3)+
1/2*k(1,3));

p(3)=feval('funccc3',t+h/2,w(1)+1/2*k(1,1),w(2)+1/2*k(1,2),w(3)+
1/2*k(1,3));
```

```
      for j=1:n
          k(2,j)=h*p(j);
      end


p(1)=feval('funccc1',t+h/2,w(1)+1/2*k(2,1),w(2)+1/2*k(2,2),w(3)+
1/2*k(2,3));

p(2)=feval('funccc2',t+h/2,w(1)+1/2*k(2,1),w(2)+1/2*k(2,2),w(3)+
1/2*k(2,3));

p(3)=feval('funccc3',t+h/2,w(1)+1/2*k(2,1),w(2)+1/2*k(2,2),w(3)+
1/2*k(2,3));
      for j=1:n
         k(3,j)=h*p(j);
      end


p(1)=feval('funccc1',t+h,w(1)+k(3,1),w(2)+k(3,2),w(3)+k(3,3));

p(2)=feval('funccc2',t+h,w(1)+k(3,1),w(2)+k(3,2),w(3)+k(3,3));

p(3)=feval('funccc3',t+h,w(1)+k(3,1),w(2)+k(3,2),w(3)+k(3,3));
      for j=1:n
         k(4,j)=h*p(j);
      end
     for j=1:n
         w(j)=w(j)+(k(1,j)+2*k(2,j)+2*k(3,j)+k(4,j))/6;
     end
         t=a+i*h;
         fprintf('\n %6.1f   %f   %f    %f ',t,w(1),w(2),w(3) );
 end



%Cada función se archiva aparte y con distinto nombre

function z=funccc1(t,y1,y2,y3) %primera función a evaluar
z=-(0.002)*y1*y2;

function z=funccc2(t,y1,y2,y3) %segunda función a evaluar
z=((0.002)*y1*y2)-((0.15)*y2);

function z=funccc3(t,y1,y2,y3) %tercera función a evaluar
z=(0.15)*y2;
```

SALIDA QUE CORRESPONDE AL CASO 2

```
RungeKuttaSEDccc

introduzca el extremo izquierdo del intervalo a:0
a =
     0
introduzca el extremo derecho del intervalo b:35
b =
    35
introduzca el numero de subintervalos m:35
m =
     35
introduzca el vector de  cond. inic. d1,d2,d3:[0.9 0.1 0]
d =
    0.9000    0.1000    0

     ti       w1i       w2i       w3i
     0.0   0.900000   0.100000   0.000000
     1.0   0.899833   0.086226   0.013941
     2.0   0.899689   0.074349   0.025962
     3.0   0.899564   0.064108   0.036328
     4.0   0.899457   0.055278   0.045265
     5.0   0.899365   0.047664   0.052972
     6.0   0.899285   0.041098   0.059617
     7.0   0.899216   0.035437   0.065346
     8.0   0.899157   0.030556   0.070287
     9.0   0.899106   0.026347   0.074547
    10.0   0.899062   0.022718   0.078220
    11.0   0.899024   0.019589   0.081387
    12.0   0.898991   0.016891   0.084118
    13.0   0.898963   0.014564   0.086473
    14.0   0.898939   0.012558   0.088503
    15.0   0.898918   0.010828   0.090254
    16.0   0.898900   0.009337   0.091764
    17.0   0.898884   0.008051   0.093065
    18.0   0.898870   0.006942   0.094188
    19.0   0.898859   0.005986   0.095156
    20.0   0.898849   0.005161   0.095990
    21.0   0.898840   0.004450   0.096710
    22.0   0.898833   0.003837   0.097330
    23.0   0.898826   0.003309   0.097865
    24.0   0.898821   0.002853   0.098326
    25.0   0.898816   0.002460   0.098724
    26.0   0.898812   0.002121   0.099067
    27.0   0.898808   0.001829   0.099363
    28.0   0.898805   0.001577   0.099618
    29.0   0.898803   0.001360   0.099837
    30.0   0.898800   0.001172   0.100027
    31.0   0.898799   0.001011   0.100190
    32.0   0.898797   0.000872   0.100331
    33.0   0.898795   0.000752   0.100453
    34.0   0.898794   0.000648   0.100558
    35.0   0.898793   0.000559   0.100648
```