UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE QUIMICA

PROGRAMACION DE METODOS NUMERICOS Y CODIFICACION EN PASCAL

T E S I S

Que para Obtener el Título de:

INGENIERO QUIMICO

P R E S E N T A

JOSE LUIS SANCHEZ LOPEZ

México, D. F. 1993

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CONTENIDO

INTRODUCCIÓN

INTRODUCCIÓN

El desarrollo de nuevos procesos en la industria química es cada vez más complejo y crecientemente más costoso. Si la investigación y el desarrollo del proceso pueden realizarse con confianza, el diseño final será más exacto, y, por tanto, la planta operará más económicamente. En todas las facetas de un proyecto de este tipo, la matemática, es el lenguaje cuantitativo, desempeña un papel vital. Por tanto un entrenamiento en métodos matemáticos es de suma importancia para los ingenieros químicos. La presente tesis se ha escrito teniendo presentes esas ideas, la intención es presentar los métodos numéricos en una forma adecuada a un ingeniero más que a un matemático. Para un matemático una prueba elegante es un fin en sí misma, pero para el ingeniero químico es meramente un medio para un fin. Por consiguiente, esta tesis muestra algunas deducciones, para dar sustento a las técnicas numéricas presentadas y así mismo dar confianza a quien las utilice para la solución de problemas de Ingeniería. Además, se muestran las técnicas más útiles de entre la amplia variedad de técnicas matemáticas disponibles.

Uno de los propósitos es inducir a los ingenieros químicos a utilizar cada vez más los métodos numéricos en la solución de sus problemas.

El contenido se desarrolla de acuerdo el siguiente esquema:

En el *capítulo I*, Generalidades, exponemos la formulación de modelos matemáticos y los tipos de modelos aplicados a la Ingeniería Química.

En el *capítulo II*, Interpolación, se exponen generalidades sobre los métodos de interpolación y abordamos el tema del trazador cúbico.

El *capítulo III*, Ecuación algebraica no lineal, nos introduce en el tratamiento de problemas no lineales. En él se revisan las principales técnicas para hallar raíces de una ecuación no lineal.

El *capítulo IV*, Integración, desarrolla los métodos para la evaluación numérica de la integral de una función sobre el intervalo en términos de diferencias calculadas con un incremento fijo.

El *capítulo V*, Ecuación diferencial ordinaria, se dedica a los métodos de resolución de una ecuación diferencial; tales métodos serán la familia de los Runge-Kutta, los métodos multipaso, y los métodos predictores-correctores.

En el *capítulo VI*, Sistemas de ecuaciones algebraicas, se exponen generalidades sobre los métodos de resolución de matrices tanto directos como iterativos. De los primeros nos dedicaremos a los métodos de Gauss y de los segundos revisaremos los métodos de Jacobi y Gauss-Seidel.

El *capítulo VII*, Ajuste de datos, nos introduce al tema del tratamiento de datos así como a la manera de representarlos mediante ecuaciones o polinomios, tocando también técnicas estadísticas.

Por último el *capítulo VIII*, Conclusiones, es un breve análisis, y recomendación sobre las técnicas numéricas y su aplicación en Ingeniería Química.

Junto con la descripción de los métodos se recogen, en todos los capítulos, algoritmos y programas que los muestran. En la mayoría de los casos no se han optimizado al máximo los algoritmos y programas (en lo que a operaciones y requerimiento de memoria se refiere), para proporcionar un mejor seguimiento de los mismos. Remitiendo al lector interesado a adaptar y optimizar los metodos aquí presentados a sus necesidades y además a consultar la bibliografía anexa para mayor comprensión de estos temas.

OBJETIVOS

OBJETIVOS

Los métodos numéricos han ido tomando una creciente importancia para el ingeniero químico. Actualmente la literatura profesional utiliza en su lenguaje común transformaciones, vectores, métodos de diferencias finitas, etc. para resolver un problema. La Ingeniería ha encontrado técnicas modernas, las cuales son herramientas invaluables en el análisis de una gran variedad de situaciones. Actualmente en la Química y en procesos industriales se tiende a dar un creciente énfasis a sistemas de control automático, simulación de procesos, diseño de equipo más eficiente, etc. Estos progresos dependen directamente de la aplicación de procedimientos numéricos que apoyados en el desarrollo tecnológico computacional, dan respuestas rápidas y confiables a una gran diversidad de problemas.

El propósito de esta tesis es, en primer lugar, consolidar el conocimiento que se tiene de los métodos numéricos adquirido durante la carrera; en segundo lugar, ofrecer estas técnicas en forma adecuada para que puedan ser aplicadas rápidamente a los problemas que se presenten tanto para el estudiante como para el profesionista de la Ingeniería Química; y tercero, presentar los métodos esenciales para resolver los problemas de Ingeniería y del área científica apoyándonos en el uso de una computadora.

Estos procedimientos están presentados con el objetivo de simplificar la comprensión de dichas técnicas, integrando una teoría adecuada en las etapas intermedias, que conduzcan a la

utilización de estos métodos numéricos en un equipo de cómputo. De esta manera el lector, ya sea estudiante o profesionista, podrá pueda apreciar el potencial de los métodos numéricos como herramienta en el área de Ingeniería, sustentada en el empleo de una computadora.

Para lograr estos propósitos, las consideraciones teóricas se complementan con la formulación de programas codificados a partir de diagramas de flujo, en lenguaje Pascal (Turbo Pascal).

El material cubierto se ubica dentro de seis grandes categorías:

Interpolación,

Ecuaciones algebraicas no lineales,

Integración,

Ecuaciones diferenciales,

Sistemas de Ecuaciones,

Ajuste de datos.

CAPITULO I

GENERALIDADES

GENERALIDADES

Los métodos numéricos, forman parte de una gran área de la Matemática, denominada análisis numérico; estos han tomado un gran auge con el avance tecnológico de las computadoras.

Los métodos numéricos generan soluciones tan aproximadas como se deseen y que puedan ser controladas según el tipo de aplicación.

Para una gran variedad de aplicaciones, el conocer el valor exacto de una cantidad suele ser innecesario. En cuestiones técnicas, por ejemplo, la cantidad buscada sirve normalmente para determinar las dimensiones u otros parámetros de un artículo manufacturado. Una gran cantidad de procesos de fabricación se basa en métodos aproximados, por lo que los cálculos técnicos cuya exactitud sobrepase las "tolerancias" permitidas, evidentemente carecerán de valor. Las fórmulas y soluciones exactas podrán ser reemplazadas por otras aproximadas, con tal de que éstas sean tan fieles a las originales que el error cometido no sobrepase ciertos límites.

De manera general el análisis o cálculo numérico actual puede entenderse como la rama de las matemáticas orientada hacia la búsqueda de algoritmos procesados por computadora. La elaboración de algoritmos exige, en primer lugar, un conocimiento profundo de las propiedades de las entidades matemáticas que queremos calcular; y es necesario controlar que el proceso en la computadora dé soluciones correspondientes a los resultados esperados.

La computadora es, en efecto, un instrumento imperfecto que puede introducir error en los cálculos, denominado "error de redondeo". La acumulación de estos puede, en ciertos casos, falsear grandemente los resultados.

Examinemos a continuación, con más detalle los requisitos para un método de cálculo. El más simple y fundamental de estos requisitos es la posibilidad de hallar la cantidad deseada con un grado prefijado de exactitud. La exactitud requerida varía mucho de un problema a otro. Para ciertos cálculos técnicos dos o tres cifras decimales serán suficientes. La mayoría de los cálculos de Ingeniería se efectúan con tres o cuatro cifras decimales. Pero la exactitud que se necesita en los cálculos científicos es mucho mayor. En general, la necesidad de exactitud ha ido aumentando con el transcurso del tiempo.

Particularmente importantes, son los métodos de aproximación y procesos que permiten obtener resultados con tanta exactitud como se desee. Tales métodos se denominan métodos de convergencia y se utilizan con mucha frecuencia.

Después de indicar cómo funcionan los métodos de aproximación, el primer problema que se plantea es el de establecer la convergencia de las aproximaciones a la solución y, si el método no siempre es convergente, señalar las condiciones bajo las cuales lo es. Una vez establecida la convergencia, surge el problema más difícil y sutil: estimar la rapidez de convergencia, es decir, con qué rapidez converge el método a la solución. Teóricamente, todo

método convergente garantiza la posibilidad de hallar la solución con cualquier grado deseado de exactitud si elegimos un grado de aproximación suficiente. Pero, por regla general, a medida que se avanza en los cálculos, más complicado es calcular la aproximación. Así, si el método converge lentamente hacia la solución, entonces obtener la exactitud necesaria puede llegar a exigir cálculos enormes.

La convergencia insuficientemente rápida es uno de los criterios por los que se juzga la desventaja de un método dado. Pero este criterio no es, naturalmente, el único y, al comparar distintos métodos, debemos considerar otros muchos aspectos, en particular el de la conveniencia de efectuar los cálculos por computadora. Entre dos métodos a veces se prefiere utilizar aquel cuya convergencia es algo más lenta, pero los cálculos son más fáciles de efectuar en una computadora.

Las necesidades de cálculo práctico imponen a los métodos de aproximación otro requisito general que se debe tener en cuenta debido a su gran importancia. Se trata de la estabilidad del proceso de cálculo. En esencia consiste en lo siguiente: cada método de aproximación conduce a un esquema de cálculo, y resulta que, para obtener todos los resultados numéricos requeridos, debemos efectuar una larga serie de pasos de acuerdo con el esquema. En cada paso el cálculo no se efectúa con exactitud sino solamente con un número determinado de cifras significativas y así, en cada paso, cometemos un pequeño error. Todos estos errores influirán en los resultados finales.

El esquema de cálculo adoptado puede resultar a veces muy poco satisfactorio, en el sentido de que los pequeños errores cometidos al principio van adquiriendo mayor influencia conforme se van efectuando los cálculos sucesivos, llegando a producir al final una gran desviación respecto a los valores exactos.

PLANTEAMIENTO MATEMÁTICO

Casi todas las ciencias aplicadas se basan en la realización de experimentos y en la interpretación de los resultados de los mismos. Esto puede hacerse cuantitativamente, efectuando medidas precisas de las variables del sistema, analizándolas y correlacionándolas, o cualitativamente, investigando el comportamiento general del sistema en función de las variables que influyen entre sí. Siempre es preferible el primer método; si se intenta realizar una investigación cuantitativa, es conveniente inducir los principios matemáticos cuanto antes, ya que pueden influir en el desarrollo de la investigación. Esto se consigue buscando un modelo matemático idealizado del sistema.

La segunda etapa consiste en la recopilación de toda la información física relevante, bajo la forma de leyes de conservación o ecuaciones de velocidad. Las leyes de conservación en Ingeniería son los balances de materia y de energía; mientras que las ecuaciones de velocidad expresan la relación entre el gasto y la fuerza impulsora en los campos de circulación de fluidos, la transmisión de calor y transferencia de masa. Estas leyes y ecuaciones se aplican entonces al modelo, y el resultado será una ecuación matemática que describirá al sistema.

El tipo de ecuación que representa al modelo (algebraica, diferencial, de diferencias finitas, etc.) dependerá tanto del sistema que se investiga como del modelo elegido. Para un sistema particular, la complejidad de la ecuación obtenida dependerá de la complejidad del modelo seleccionado. Se aplican entonces a esta ecuación las técnicas numéricas apropiadas y se obtiene un resultado. Este resultado deberá interpretarse posteriormente de acuerdo al modelo original, con el objeto de que tenga significado físico.

ERRORES

El análisis del error en un resultado numérico es fundamental para cualquier cálculo, sea hecho a mano o mediante una computadora. Los datos de entrada rara vez son exactos, ya que a menudo se basan en experimentos o en estimaciones; los procesos numéricos, a su vez, introducen errores de varios tipos. Dependiendo de la fuente que los produzca, existen varios tipos de errores. Los errores básicos en los que se incurre son:

Inherentes

Son errores que existen en los valores de los datos, originados por incertidumbre en las mediciones, a causa de equivocaciones en la lectura o por la naturaleza necesariamente aproximada de la lectura de cantidades que no pueden representarse exactamente con el número de dígitos necesario para trabajar.

Truncación

Estos son ocasionados por el método numérico en sí, el nombre surge porque muchos de los métodos son comparados con la serie de Taylor truncada. En general, al realizar cálculos utilizando series con un número infinito de términos, se tiene que omitir parte de éstos para realizar las operaciones, con lo cual introduce un error causado por los términos truncados.

Redondeo

Todos los dispositivos de cálculo representan números con alguna imprecisión. En las computadoras las cantidades numéricas se manejan como números de punto flotante con una magnitud de palabra fija. De ahí la imposibilidad de manejar en operaciones aritméticas todos los dígitos resultantes que involucran estas operaciones, en este caso el resultado se redondea al número máximo de dígitos de los que se dispone.

Existen otros tipos de error, como el error de propagación que es consecuencia de uno o varios de los anteriores al presentarse en procesos sucesivos de cálculo.

La medición del error generado se puede cuantificar mediante el error absoluto o el error relativo. El error absoluto se define como el valor absoluto de la diferencia entre un valor cualquiera x y el valor estimado de éste representado por x_1.

$$e_a = x - x_1$$

el error relativo se define como el valor absoluto del cociente de la diferencia entre un valor cualquiera de x y el valor estimado de éste dividido entre el valor de x y expresado en porciento.

$$e_r = \left| \frac{x - x_1}{x} \right| * 100$$

ESTABILIDAD

Los errores provenientes de cualquier fuente se propagan de diferentes formas, algunos crecerán poco y por lo tanto no afectarán en forma significativa la exactitud de el resultado, otros pueden crecer tanto que invaliden completamente los resultados.

CONVERGENCIA

El cálculo y generalmente el análisis están basados en este concepto. Así, la derivada, la integral y la continuidad se definen en términos de sucesiones convergentes. En Ingeniería no se requiere de resultados numéricos exactos, sino más bien de una aproximación a la respuesta hasta un cierto número de cifras decimales o hasta estar dentro de una tolerancia. En muchos métodos numéricos se producen sucesiones de valores, esto es, soluciones aproximadas que convergen a la solución, si para todo $\epsilon > 0$ existe un entero m tal que para todo $n > m$; se cumple que

$$| x - x_m | < \epsilon$$

entonces la solución converge a la respuesta y podremos obtener a x con cualquier exactitud.

ALGORITMO

Un algoritmo es una lista de instrucciones paso por paso para llevar a cabo algún proceso. Una técnica de laboratorio sería un ejemplo de algoritmo: la síntesis de algún compuesto aún cuando es un proceso más complicado, puede subdividirse en pasos sencillos y comprensibles para alguien con experiencia en Química y también sería un ejemplo de algoritmo.

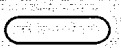
En la misma forma, los algoritmos ejecutados por una computadora pueden combinar, en un cálculo matemático complicado, millones de pasos elementales, tales como adiciones y substracciones. Una computadora puede también, mediante el uso de algoritmos adecuados, controlar un proceso de manufactura o coordinar las reservaciones de una línea aérea en todo el mundo. Los algoritmos para procesos a gran escala son más complicados, pero están construidos a base de elementos más sencillos.

DIAGRAMA DE FLUJO

Tal vez la técnica de representación de algoritmos más popular sea el empleo de diagramas de flujo. Estos diagramas ilustran el flujo de datos e información así como el manejo de éstos mediante símbolos interconectados con líneas. La combinación de símbolos y líneas representa la lógica del programa. En la siguiente página se muestran los símbolos más comúnmente utilizados en diagramas de flujo.

SIMBOLOGIA

DIGRAMAS DE FLUJO

	INICIO/TERMINO DE PROGRAMA		CONECTOR DE HOJA
	LECTURA DE DATOS		CONECTOR DE PAGINA
	PROCESO CICLICO		IMPRESION DE DATOS
	CONDICIONAL DECISION		LINEAS DE FLUJO
	PROCESO		SUBPROGRAMA

Los símbolos indican el tipo de operación que se va a realizar y la secuencia de la operación que deba realizarse.

- Un rectángulo significa algún tipo de procedimiento. El proceso podría ser tan específico como "calcular la calificación promedio de un individuo" o tan general como "preparar la programación cronológica de clases para el semestre".
- El trapezoide representa un proceso de entrada que denota cualquier tipo de alimentación de datos al programa.
- El rombo marca el punto donde debe realizarse alguna decisión. Con base en esta decisión la secuencia de ejecución seguirá hacia un conjunto de operaciones.
- Terminal, cada diagrama de flujo debe empezar y terminar con un símbolo de este tipo.
- Un círculo pequeño, es un conector de hoja dentro de la misma página y se emplea para cortar y después enlazar líneas de flujo en la misma página.
- Conector de página o fuera de página es un símbolo pentagonal, éste realiza una función semejante al anterior cuando las líneas de flujo deben extenderse de una página a otra.
- Proceso predefinido, se representa mediante un rectángulo con líneas verticales adicionales, identifica a aquellos procesos que son estándar. Por lo general estos procesos se definen en alguna otra parte, generalmente en algún diagrama de flujo separado.
- Salida a impresora indica que el contenido será desplegado en la impresora o en el monitor.

Existen además otros símbolos más específicos para diferentes tareas. Todos estas figuras representan la lógica del programa. El diagrama de flujo en sí es un esquema que representa a un algoritmo.

RESOLUCIÓN DEL PROBLEMA Y PROGRAMACIÓN

Un programa es un conjunto de actividades codificadas en algún lenguaje de programación que corresponden a la realización de una tarea. En el contexto de la programación, el dar solución a un problema involucra varias etapas, cada una de las cuales tiene sus propios aspectos y dificultades. Estas etapas son:

Definición del problema.

Proposición de una solución.

Definición de requerimiento de datos.

Desarrollo del algoritmo.

Codificación del algoritmo.

Evaluación del resultados.

Documentación del programa.

Mantenimiento.

MÉTODOS NUMÉRICOS

El advenimiento de las computadoras en las décadas de los 40's y los 50's fue debido a los grandes esfuerzos de científicos que requerían dar solución a problemas precisando un alto grado de exactitud en la solución. Así, para el uso de la computadora, el científico tiene que expresar sus modelos matemáticos y soluciones en forma numérica. La rama la Matemática que se ocupa de esto es el análisis numérico y de éste obtenemos los métodos numéricos.

Un método numérico es una expresión matemática de una solución que produce una respuesta numérica. Un matemático, por lo general, maneja expresiones matemáticas y también produce respuestas de ese tipo, no así el ingeniero a quién le interesa obtener resultados numéricos con el fin de analizarlos para encontrar las relaciones existentes entre las variables estudiadas.

Las técnicas matemáticas pueden ser expresadas numéricamente incluyendo:

Diferenciación,

Integración,

Localización de raíces de ecuaciones,

Solución de ecuaciones diferenciales,

Solución de ecuaciones simultáneas,

Ajuste de curvas e interpolación,

y Manejo de matrices.

La importancia de los métodos numéricos reside en que estos se aplican también donde la respuesta analítica es difícil de encontrar.

CAPITULO II

INTERPOLACION

INTERPOLACIÓN

Frecuentemente, el ingeniero tiene que enfrentarse con un conjunto de datos que relacionan dos variables, tales como la presión y temperatura que reinan en el interior de un recipiente. La presión es una función desconocida de la temperatura, y se desea calcular la presión correspondiente a cierta temperatura dentro o fuera de los límites de los datos disponibles. El procedimiento matemático se denomina interpolación, cuando la presión que se desea conocer está dentro de los límites de los datos extremos; mientras que, si el punto cae fuera del intervalo de los datos, el procedimiento matemático se denomina extrapolación.

La interpolación y la extrapolación se basan en la suposición de que la relación funcional entre las dos variables es continua en el intervalo de la variable independiente que se considera. Por tanto para interpolar o extrapolar, el ingeniero debe obtener esta relación funcional u otra relación aproximada, para ello debe considerarse la tendencia de los datos desde el punto de vista de los resultados experimentales y del análisis matemático. Por ejemplo, el ingeniero químico debe estar familiarizado con la forma general de una curva presión de vapor-temperatura, y con su situación en un gráfico a partir de datos experimentales. El matemático es capaz de asignar una ecuación aproximada a esa curva, y entonces la interpolación no tiene ninguna complicación.

Habitualmente los datos experimentales se correlacionan mediante un polinomio, y el grado del polinomio puede estimarse preparando una tabla de diferencias a partir de los datos recogidos. La columna de diferencias que dá un valor aproximadamente constante proporciona el grado del polinomio al que pueden ajustarse los datos por selección de las constantes.

INTERPOLACIÓN DE LAGRANGE

Como se ha visto, se puede aproximar el comportamiento de una serie de datos mediante una función o polinomio de aproximación. Muy frecuentemente, los intervalos de la variable independiente en los datos disponibles son desiguales, lo que elimina la posibilidad de utilizar una tabla de diferencias. Además, en la vida real no siempre se puede obtener un muestreo de datos muy estricto debido a que en algunos casos es excesivamente costoso mantener el control, en otros debido a que en el experimento realizado tenemos que desechar ciertas lecturas que no garanticen apego a la realidad o bien que se sospeche de algún sesgo.

La fórmula de Lagrange para el polinomio de interpolación esta dado por

$$P_n(x) = \sum_{i=0}^{n} L_i(x)\, f(x_i)$$

donde $L_i(x)$ es la función multiplicadora de Lagrange

$$L_i(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}$$

de otra manera

$$L_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{(x-x_j)}{(x_i-x_j)}, \qquad i = 0,1,\ldots,n$$

cada valor funcional $f(x_i)$ esta multiplicado por el polinomio $L_i(x)$ un polinomio de grado n en x (ya que existen factores $(x-x_i)$).

La fórmula de Lagrange se puede obtener directamente a partir del polinomio de Newton de grado equivalente escribiendo primeramente los cocientes incrementales en forma simétrica. Consideremos por ejemplo el polinomio en coeficientes incrementales de segundo grado.

$$f(x) = P_n(x) + R_n(x)$$
$$P_2(x) = f[x_o] + (x-x_o)[x_1, x_o] + (x-x_o)(x-x_1)[x_2, x_1, x_o]$$

substituyendo las formas simétricas de los cocientes incrementales equivalentes se deduce

$$P_2(x) = f(x_o) + (x-x_o)\frac{f(x_o)}{(x_o-x_1)} + (x-x_o)\frac{f(x_1)}{(x_1-x_o)}$$

$$+ \frac{(x-x_o)(x-x_1)}{(x_o-x_1)(x_o-x_2)} f(x_o) + \frac{(x-x_o)(x-x_1)}{(x_1-x_o)(x_1-x_2)} f(x_1) + \frac{(x-x_o)(x-x_1)}{(x_2-x_o)(x_2-x_1)} f(x_2)$$

$$= \frac{(x-x_1)(x-x_2)}{(x_o-x_1)(x_o-x_2)} f(x_o) + \frac{(x-x_o)(x-x_2)}{(x_1-x_o)(x_1-x_2)} f(x_1) + \frac{(x-x_o)(x-x_1)}{(x_2-x_o)(x_2-x_1)} f(x_2)$$

$$= f(x_o)\left[\prod_{\substack{j=o \\ j\neq o}}^{2} \frac{(x-x_j)}{(x_o-x_j)}\right] + f(x_1)\left[\prod_{\substack{j=o \\ j\neq 1}}^{2} \frac{(x-x_j)}{(x_1-x_j)}\right] + f(x_2)\left[\prod_{\substack{j=o \\ j\neq 2}}^{2} \frac{(x-x_j)}{(x_2-x_j)}\right]$$

$$P_2(x) = \sum_{i=o}^{2} f(x_i)\left[\prod_{\substack{j=o \\ j\neq i}}^{2} \frac{(x-x_j)}{(x_i-x_j)}\right]$$

$$P_2(x) = \sum_{i=o}^{2} L_i(x) f(x_i)$$

Algoritmo

Para interpolar algún valor a partir de una tabla de datos:

Entrada n número de datos, m grado del polinomio,

x_i, $f(x_i)$ valor iésimo del argumento y de la función,

x valor a interpolar.

Salida x valor a interpolar, f(x) aproximación interpolada.

Paso 1 para i = 1, .., n-1 hacer 2

Paso 2 Si ($x \geq x_i$ y $x \leq x_{i+m}$) y ($x \leq x_i$ y $x \geq x_{i-m}$) entonces Paso 3

sino Paso 6

Paso 3 r = i

Paso 4

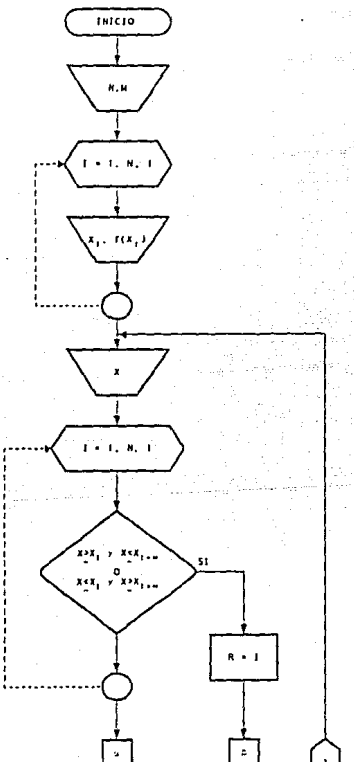
$$f(x) = \sum_{i=r}^{r+m} f(x_i) \left[\prod_{\substack{k=r \\ i \neq k}}^{r+m} \frac{(x - x_k)}{(x_i - x_k)} \right]$$

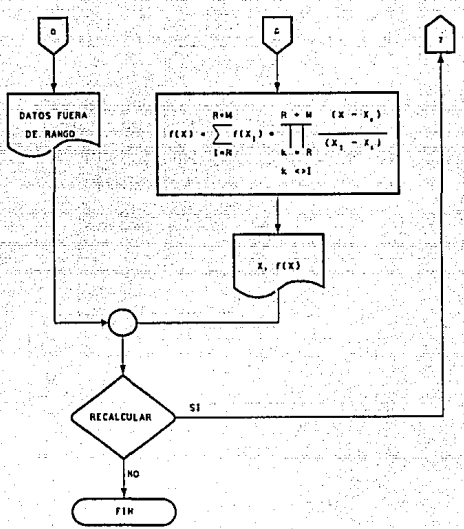
Paso 5 Salida (x, f(x)). Terminar.

Paso 6 Salida (Datos fuera de rango)

Terminar.

INTERPOLADOR DE LAGRANGE




0
DATOS FUERA DE RANGO
f(X) = Σ f(X_I) = Π (X − X_k)/(X_I − X_k)
I=R
R+M
k = R
k <> I
C
X, f(X)
RECALCULAR
SI
NO
FIM
7

```
Program Interpolacion_de_Lagrange;
uses Crt;
var g, i, j, n, m, r, mensaje : Integer;
    Inter, producto, xi : real;
    x,fx: array[1..20] of real;
    c : char;

Procedure Inicializa;
begin
  i := 0;
  j := 0;
  n := 0;
  m := 0;
  r := 0;
  mensaje := 0;
  Inter := 0;
  producto := 1;
  xi := 0;
  c := char(0);
end;

procedure Captura;
begin
  clrscr;
  gotoxy(28,2);write('INTERPOLACION DE LAGRANGE');
  gotoxy(22,4);write('interpolación con espacios variables.');
  gotoxy(29,6);write('Suministre lo siguiente');
  gotoxy(29,8);write('Número de datos:  n = ');
  gotoxy(51,8);read(n);
  for i := 1 to n do
    begin
      gotoxy(24,9+i);write('x',i,' = ');
      gotoxy(48,9+i);write('f(x',i,') = ');
    end;
  gotoxy(25,21);write('Valor de x a interpolar: x = ');
  gotoxy(21,24);write('Presione cualquier tecla para continuar');
  for i := 1 to n do
    begin
      gotoxy(30,9+i);read(x[i]);
      gotoxy(56,9+i);read(fx[i]);
    end;
  gotoxy(54,21);read(xi);
  gotoxy(60,24);c := readkey;
end;
```

```
  procedure Lagrange;

  procedure Interpolacion;
  begin
    for i := 1 to n do
      begin
        producto := 1;
        for j := 1 to n do
          if j <> i then
            producto := producto * (xi - x[j])/(x[i]-x[j]);
        inter := inter + Fx[i] * producto;
      end;
    mensaje := 1;
  end;

  begin
    for i := 1 to n do
      if ((xi >= x[i]) and (xi <= x[n])) or
         ((xi <= x[1]) and (xi >= x[n])) then Interpolacion
                                         else mensaje := 2;
  end;

  procedure Resultado;
  begin
    if mensaje = 1 then
      begin
        gotoxy(1,21);clreol;
        gotoxy(34,21);write('Interpolación');
        gotoxy(24,22);write('x = ',xi:5:8,'  f(x) =',Inter:5:8);
      end
    else
      begin
        gotoxy(1,21);clreol;
        gotoxy(34,21);write('Interpolación');
        gotoxy(25,22);write('Dato a interpolar fuera de rango');
      end;
    gotoxy(60,24);c := readkey;
  end;

  begin
    Inicializa;
    Captura;
    Lagrange;
    Resultado;
  end.
```

INTERPOLACIÓN CUBICA POR SEGMENTOS

En lugar de utilizar un sólo polinomio de alto grado que represente a una función sobre un intervalo, podemos emplear polinomios conjuntados, cada uno de ellos de bajo grado. Por ejemplo segmentos de línea, en donde cada uno de ellos corresponde a los datos proporcionados sobre un subintervalo. Tal aproximación es continua pero tiene una primera derivada con discontinuidades en los extremos del intervalo, como se ve en la gráfica 1 de la página 35.

Consideremos ahora un método en el que se construyen segmentos cúbicos de tal manera que las esquinas se redondean, siendo continuas tanto la primera como la segunda derivada de la aproximación. Los polinomios de alto grado tienen características oscilatorias. Uno de grado n puede tener tantos como n-1 puntos de retorno. Cuando un polinomio de tales características representa con precisión una función dada, ello suele ser por una oscilación regresiva y progresiva a través de la función. Esto tiene efectos colaterales indeseables, por ejemplo una pobre aproximación de la derivada, por mencionar sólo uno. La aproximación por interpolación por segmentos que se obtendrá ahora, evita dichas oscilaciones porque está compuesta de segmentos de bajo grado.

Dado un intervalo (a,b) = I dividido en n subintervalos por los puntos $x_0 = a$, x_1, x_2, ..., $x_n = b$, un segmento cúbico se ajustará a cada subintervalo, tomando valores específicos de y_i en los puntos x_i, con la primera y segunda derivadas en subintervalos adyacentes que

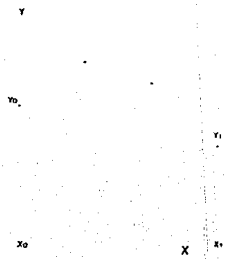
convergen en el valor con la unión. Los puntos x_1 a x_{n-1} se llaman nodos, o nudos de la interpolación por segmentos, esto se muestra en la gráfica II de la siguiente página.

Resumiendo, la interpolación por segmentos es una aplicación sofisticada de la interpolación, ya que en lugar de emplear un sólo polinomio de aproximación, se emplean varios polinomios unidos, debido a que se crea un polinomio de bajo grado entre cada uno de los intervalos de la muestra, además de que se reducen los picos. Otra ventaja es que al emplear polinomios de bajo grado evitamos posibles oscilaciones que ocurrirían con polinomios de alto grado.

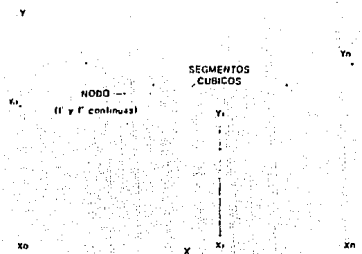
Como todos los casos de aproximación polinomial, una vez que tenemos la aproximación adecuada, podemos derivarla, integrarla, evaluarla, conocer su comportamiento, obtener sus raíces y en general emplearla para hacer cualquier tipo de operaciones que requiramos.

TRAZADOR CUBICO POR SEGMENTOS

GRAFICA I



GRAFICA II



Algoritmo:

Construcción del trazador cúbico TC para la función f, definida en los números $x_0 < x_1 < \ldots < x_n$, que satisface $TC'(x_0)=f'(x_0)$ y $TC'(x_n)=f'(x_n)$:

Entrada n, $(x_i, f(x_i))$ para $i=1, \ldots, n$; $f'(x_0)$, $f'(x_n)$, NI, XI_i para $i=1, .., NI$

Salida a_i, b_i, c_i, d_i, para $i=1, \ldots, n-1$; $f'(x_0)$, $f'(x_n)$

x_i, $f(x_i)$ para $i=1, \ldots, n$; XI_i para $i=1, \ldots, NI$

Paso 1 para $i = 1, \ldots, n-1$ hacer $INTE_i = x_{i+1} - x_i$

Paso 2 para $i = 1, \ldots, n$ hacer $a_i = y_i$

Paso 3 $\alpha_1 = 3(a_1 - a_0)/INTE_1 - 3DI$

$\alpha_n = 3DI - 3(a_n - a_{n-1})/INTE_{n-1}$

Paso 4 para $i = 2, \ldots, n-1$

$$\alpha_i = \frac{3\,(a_{i+1} \cdot INTE_{i-1} - a_i\,(X_{i+1} - X_{i-1}) + a_{i-1} \cdot INTE_i)}{INTE_{i-1} \cdot INTE_i}$$

Paso 5 $L_1 = 2INTE_1$

$\mu_1 = 0.5$

$z_1 = a_1/L_1$

Paso 6 para $i = 2, \ldots, n-1$

$L_i = 2(X_{i+1} - X_{i-1}) - INTE_{i-1} * \mu_{i-1}$

$\mu_i = INTE_i/L_i$

$z_i = (\alpha_i - INTE_{i-1} * z_{i-1})/L_i$

Paso 7 $C_n = (\alpha_n - INTE_{n-1} * z_{n-1}) / (INTE_{n-1} * (2 - \mu_{n-1}))$

Paso 8 para $i = n-1, \ldots, 1$

$$C_i = z_i - \mu_i * C_{i+1}$$

Paso 9 para i = n-1, ... 1

$$B_i = \frac{a_{i+1} - a_i}{INTE_i} - INTE_i \frac{C_{i+1} + 2C_i}{3}$$

$$D_i = \frac{C_{i+1} - Ci}{3INTE_i}$$

Paso 10 para i = 1, ..., NI

Lugar = 1

para t = 1, ..., n-1

if XI > X_t entonces Lugar = t

$X = XI_i - X_t$

$YI_i = D_{Lugar}X + C_{Lugar}X + B_{Lugar}X + A_{Lugar}$

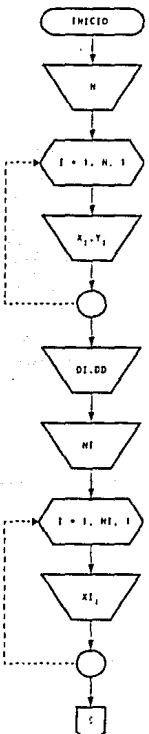
Paso 11 Salida (para i = 1, ..., n: X_i, Y_i, XI_i

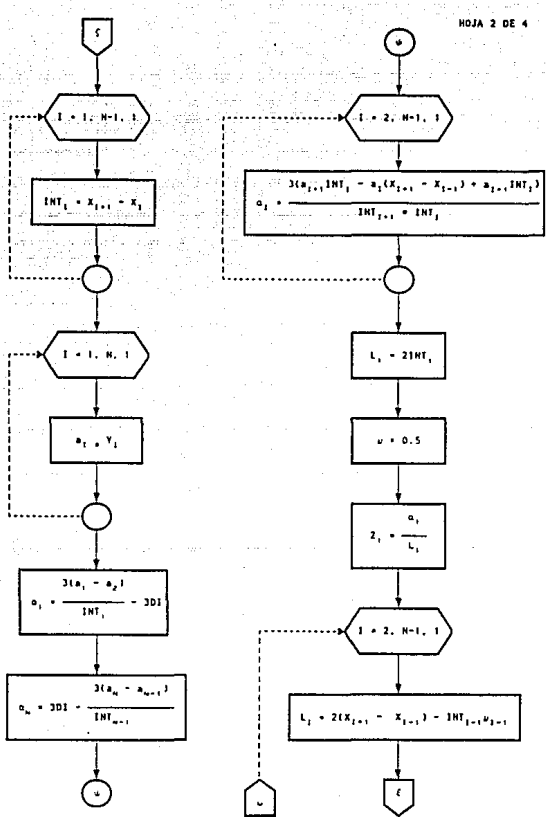
DD, DI

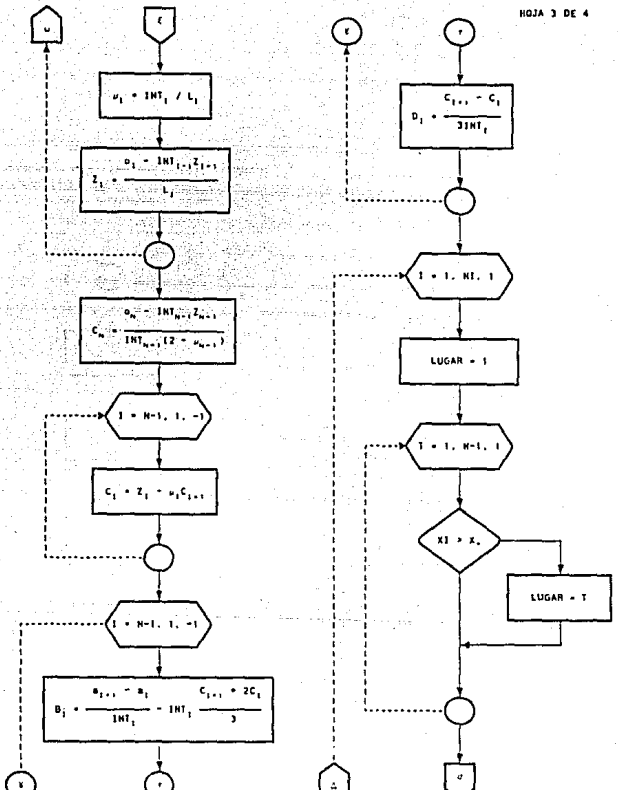
para i = 1, ..., n-1: a_i, b_i, c_i, d_i)

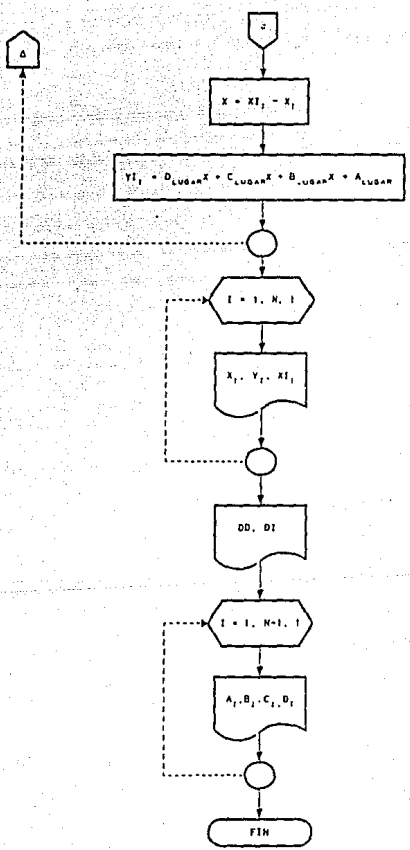
Termina.

INTERPOLADOR CUBICO POR SEGMENTOS




5
I = 1, N-1, 1
INT_I = X_{I+1} - X_I
I = 1, N, 1
a_I = Y_I
a_1 = 3(a_1 - a_2)/INT_1 - 3DI
a_N = 3DI - 3(a_N - a_{N-1})/INT_{N-1}
4
4
I = 2, N-1, 1
a_I = (3(a_{I+1}INT_I - a_I(X_{I+1} - X_{I-1}) + a_{I-1}INT_I)) / (INT_{I-1} INT_I)
L_1 = 2INT_1
w = 0.5
Z_1 = a_1 / L_1
I = 2, N-1, 1
L_I = 2(X_{I+1} - X_{I-1}) - INT_{I-1}w_{I-1}
w
6


U
C
$\mu_1 = INT_1 / L_1$
$Z_1 = \frac{a_1 - INT_{1-1} Z_{1+1}}{L_1}$
$C_N = \frac{a_N - INT_{N-1} Z_{N+1}}{INT_{N-1} (2 - \mu_{N-1})}$
I = N-1, 1, -1
$C_1 = Z_1 - \mu_1 C_{1+1}$
I = N-1, 1, -1
$B_1 = \frac{a_{1+1} - a_1}{INT_1} - INT_1 \frac{C_{1+1} + 2C_1}{3}$
V
Y
E
Y
$D_1 = \frac{C_{1+1} - C_1}{3INT_1}$
I = 1, NI, 1
LUGAR = 1
I = 1, N-1, 1
X1 > X.
LUGAR = I
A
Q


C
A
X = XI2 - X1
YI1 = DLUGARX + CLUGARX + BLUGARX + ALUGAR
I = 1, N, 1
X1, Y1, XI1
DD, DI
I = 1, N-1, 1
A1, B1, C1, D1
FIN

```
Program Interpolador_Cubico_por_Segmentos;
uses crt;

{$I-}
{$R+}

const TamArr = 50;
      ErrES : boolean = false;
type  vector = array[0..TamArr] of real;
var   NumPunts, NumPtsInter  : integer;
      DerivFI, DerivFD : real;
      Coef0, Coef1, Coef2, Coef3, DatoX, DatoY : vector;
      XInter, YInter : vector;
      Error, Pantalla : byte;
      ArchSal : text;
      Cont : char;

procedure TrazadorSeg(NumPunts : integer;
          var DatoX, DatoY   : vector;
          DerivFI, DerivFD   : real; NumPtsInter  : integer;
          var XInter, Coef0, Coef1, Coef2, Coef3, YInter : vector;
          var Error      : byte);
{$R+}
const CasiCero = 1E-015;
type  coeficiente = record
                      A, B, C, D : vector;
                    end;
var Intervalo: vector;
    Trazador : coeficiente;

procedure CalculaIntervalos(NumPunts : integer;
          var DatoX, Intervalo : vector;
          var Error: byte);
var Indice : integer;
begin
  Error := 0;
  for Indice := 1 to NumPunts - 1 do
  begin
    Intervalo[Indice] := DatoX[Indice+1] - DatoX[Indice];
    if ABS(Intervalo[Indice]) < CasiCero then
       Error := 1;
    if Intervalo[Indice] < 0 then
       Error := 2;
  end;
end;

procedure CalculaCoeficientes(NumPunts : integer;
          var DatoX, DatoY, Intervalo : vector;
          DerivFI, DerivFD : real;
          var Trazador : coeficiente);
```

```
procedure CalculaAs(NumPunts : integer; var DatoY : vector;
          var Trazador : coeficiente);
var
  Indice : integer;
begin
  for Indice := 1 to NumPunts do
    Trazador.A[Indice] := DatoY[Indice];
end;

procedure CalculaCs(NumPunts : integer;
          var DatoX, Intervalo : vector;
          DerivFI, DerivFD : real; var Trazador : coeficiente);
var Alfa, L, Mu, Z : vector;
    Indice : integer;
begin
  with Trazador do
  begin
    Alfa[1] := 3*(A[2] - A[1]) / Intervalo[1] - 3 * DerivFI;
    Alfa[NumPunts] := 3 * DerivFD - 3
                    * (A[NumPunts] - A[NumPunts - 1])
                    / Intervalo[NumPunts - 1];
    for Indice := 2 to NumPunts - 1 do
      Alfa[Indice] := 3 * ((A[Indice+1] * Intervalo[Indice-1])
                      - (A[Indice] * (DatoX[Indice+1]
                      - DatoX[Indice-1]))
                      + (A[Indice-1] * Intervalo[Indice]))
                      / (Intervalo[Indice-1]
                      * Intervalo[Indice]);
    L[1] := 2 * Intervalo[1];
    Mu[1] := 0.5;
    Z[1] := Alfa[1] / L[1];
    for Indice := 2 to NumPunts - 1 do
    begin
      L[Indice] := 2 * (DatoX[Indice+1] - DatoX[Indice-1]) -
                Intervalo[Indice-1] * Mu[Indice-1];
      Mu[Indice] := Intervalo[Indice] / L[Indice];
      Z[Indice] := (Alfa[Indice] - Intervalo[Indice-1]
                  * Z[Indice-1]) / L[Indice];
    end;
    C[NumPunts] := (Alfa[NumPunts] -
                Intervalo[NumPunts - 1] * Z[NumPunts - 1])
                / (Intervalo[NumPunts - 1]
                * (2-Mu[NumPunts - 1]));
    for Indice := NumPunts - 1 downto 1 do
      C[Indice] := Z[Indice] - Mu[Indice] * C[Indice+1];
  end;
end;
```

```
procedure CalculaDs(NumPunts : integer; var Interval : vector;
                    var Trazador : coeficiente);
var Indice : integer;
begin
  with Trazador do
    for Indice := NumPunts - 1 downto 1 do
    begin
      B[Indice] := (A[Indice+1] - A[Indice])
                   / Intervalo[Indice] - Intervalo[Indice]
                   * (C[Indice+1] + 2 * C[Indice]) / 3;
      D[Indice] := (C[Indice+1] - C[Indice])
                   / (3 * Intervalo[Indice]);
    end;
end;

begin
  CalculaAs(NumPunts, DatoY, Trazador);
  CalculaCs(NumPunts, DatoX, Intervalo, DerivFI,
            DerivFD, Trazador);
  CalculaDs(NumPunts, Intervalo, Trazador);
end;

procedure Interpolacion(NumPunts : integer; var DatoX : vector;
          var Trazador : coeficiente; NumPtsInter : integer;
          var XInter, YInter : vector);
var Indice, Ubicacion, Term : integer;
    X : real;
begin
  for Indice := 1 to NumPtsInter do
  begin
    Ubicacion := 1;
    for Term := 1 to NumPunts - 1 do
      if XInter[Indice] > DatoX[Term] then
        Ubicacion := Term;
    X := XInter[Indice] - DatoX[Ubicacion];
    with Trazador do
      YInter[Indice] := ((D[Ubicacion] * X + C[Ubicacion]) * X
                        + B[Ubicacion]) * X + A[Ubicacion];
  end;
end;

begin
  Error := 0;
  if NumPunts < 2 then
    Error := 3
  else
    CalculaIntervalos(NumPunts, DatoX, Intervalo, Error);
```

```
  if Error = 0 then
  begin
    CalculaCoeficientes(NumPunts, DatoX, DatoY, Intervalo,
                        DerivFI, DerivFD, Trazador);
    Interpolacion(NumPunts, DatoX, Trazador, NumPtsInter,
                  XInter, YInter);
  end;
  Coef0 := Trazador.A;
  Coef1 := Trazador.B;
  Coef2 := Trazador.C;
  Coef3 := Trazador.D;
end;

procedure DespliegaErr;
begin
  Writeln(ArchSal, '          !! ---------- !!');
  Write(ArchSal, '          !!          ');
  LowVideo;
  Write(ArchSal, 'ERROR       ');
  HighVideo;
  Writeln(ArchSal, '!!');
  Writeln(ArchSal, '          !! ---------- !!');
  Writeln(ArchSal);
end;

procedure RevisaES;
const Bell = #7;
type Prompt = string[80];
var IOcode : integer;

procedure Error(Msg : Prompt);
begin
  Writeln;
  Writeln(Bell, Msg);
  Writeln;
end;

begin
  IOcode := IOresult;
  ErrES := IOcode <> 0;
  if ErrES then
    case IOcode of
      $01 : Error('Archivo no existente');
      $02 : Error('Archivo no abierto para entrada.');
      $03 : Error('Archivo no abierto para salida.');
      $04 : Error('Archivo no abierto.');
      $10 : Error('Error en el formato numérico.');
      $22 : Error('Asignación a archivos estándar no permitida.');
      $91 : Error('Búsqueda después del fin de archivo.');
      $99 : Error('Fin de archivo inesperado.');
```

```
    $F0 : Error('Error escritura en disco.');
    $F1 : Error('Directorio lleno.');
    $F3 : Error('Revise FILES=20 en el CONFIG.SYS.');
   else
    begin
     Writeln;
     Writeln(Bell);
     Writeln('Mensaje de error indefinido = ', IOcode, '.');
     Writeln;
    end;
  end;
end;

procedure Encabezado;
begin
 Clrscr;
 Gotoxy(26,3);Write('TRAZADOR CUBICO POR SEGMENTOS');
 Gotoxy(25,6);Write('Suministre los Siguientes datos');
end;

procedure ObtenarchSalida(var ArchSal : text);
var NombreArch : string[255];
    C : char;
begin
 Clrscr;
 Encabezado;
 Gotoxy(20,6);Write('Salida directa a uno de los siguientes: ');
 Gotoxy(35,10);Write('(P)antalla.');
 Gotoxy(35,13);Write('(I)mpresora.');
 Gotoxy(35,16);Write('(A)rchivo.');
 Gotoxy(35,20);Write('Opción  [_]');
 Gotoxy(21,24);Write('Presione cualquier tecla para continuar');
 repeat
  Gotoxy(44,20);Readln(C);
  C := UpCase(C);
 until(C in ['P', 'I', 'A']);
 case C of
  'P' : begin
         NombreArch := 'CON';
         Pantalla := 1;
        end;
  'I' : NombreArch := 'PRN';
  'A' : begin
         repeat
          Encabezado;
          C := 'S';
          Gotoxy(14,9);Write('Inserte el nombre del archivo');
          Gotoxy(31,11);Write('Nombre: ');
          Gotoxy(21,24);
          Write('Presione cualquier tecla para continuar');
```

```
            Gotoxy(40,11);Readln(NombreArch);
            Assign(ArchSal, NombreArch);
            Reset(ArchSal);
            if IOresult = 0 then
            begin
              Close(ArchSal);
              Gotoxy(29,15);Write('Este archivo ya existe.');
              Gotoxy(28,18);Write('Subreescribir (s/n):  [_]');
              Gotoxy(51,18);Read(C);
              C := UpCase(C);
            end;
            if C = 'S' then
            begin
              Rewrite(ArchSal);
              RevisaES;
            end;
          until((C = 'S') and not(ErrES));
        end;
  end;
  Assign(ArchSal, NombreArch);
  Rewrite(ArchSal);
end;

procedure Inicializa(var Coef0, Coef1, Coef2, Coef3 : vector;
        var DatoX, DatoY : vector;
        var DerivF1, DerivFD : real;
        var NumPunts, NumPtsInter: integer;
        var XInter, YInter : vector; var Error : byte);
begin
  FillChar(Coef0, SizeOf(Coef0), 0);
  FillChar(Coef1, SizeOf(Coef1), 0);
  FillChar(Coef2, SizeOf(Coef2), 0);
  FillChar(Coef3, SizeOf(Coef3), 0);
  FillChar(DatoX, SizeOf(DatoX), 0);
  FillChar(DatoY, SizeOf(DatoY), 0);
  FillChar(XInter, SizeOf(XInter), 0);
  FillChar(YInter, SizeOf(YInter), 0);
  Pantalla := 0;
  NumPunts := 0;
  NumPtsInter := 0;
  DerivF1 := 0;
  DerivFD := 0;
  Error := 0;
  Writeln;
end;
```

```
procedure Captura(var NumPunts : integer;
          var DatoX, DatoY : vector;
          var DerivFI, DerivFD : real;
          var NumPtsInter : integer;
          var XInter : vector);
var C : char;


procedure Obten2VecsDeTeclado(var NumPunts : integer;
          var DatoX, DatoY : vector);
var Term : integer;
begin
  NumPunts := 0;
  Writeln;
  repeat
    Gotoxy(25,10);Write('Número de puntos (0-', TamArr,') n = ');
    Readln(NumPunts);
    RevisaES;
  until ((NumPunts >= 0) and (NumPunts <= TamArr) and not ErrES);
  for Term := 1 to NumPunts do
    begin
      Gotoxy(25,11+Term);Write('X[',Term,'] = ');
      Gotoxy(45,11+Term);Write('Y[',Term,'] = ');
    end;
  Gotoxy(21,24);Write('Presione cualquier tecla para continuar');
  for Term := 1 to NumPunts do
  repeat
    Gotoxy(33,11+Term);Read(DatoX[Term]);
    Gotoxy(53,11+Term);Read(DatoY[Term]);
    RevisaES;
  until not ErrES;
  Gotoxy(60,24);Cont := readkey;
end;

procedure Obten1VecDeTeclado(var NumPtsInter : integer;
          var XInter : vector);
var Term : integer;
begin
  NumPtsInter := 0;
  repeat
    Gotoxy(1,8);ClrEol;
    Gotoxy(25,8);Write('Número de puntos (0-',TamArr,'): n = ');
    Gotoxy(55,8);Readln(NumPtsInter);
    RevisaES;
  until((NumPtsInter >= 0) and (NumPtsInter <= TamArr) and not ErrES);
  Writeln;
  for Term := 1 to NumPtsInter do
    begin
      Gotoxy(35,10+Term);Write('Punto ',Term,': ');
    end;
```

```
  Gotoxy(21,24);Write('Presione cualquier tecla para continuar');
  for Term := 1 to NumPtsInter do
  repeat
    Gotoxy(45,10+Term);ReadIn(XInter[Term]);
    RevisaES;
  until not ErrES;
  Gotoxy(60,24);Cont := readkey;
end;
begin
  Encabezado;
     Obten2VecsDeTeclado(NumPunts, DatoX, DatoY);
     Encabezado;
     Gotoxy(10,11);Write('Derivada de la función:');
     Gotoxy(25,15);Write('En el punto final izquierdo: ');
     Gotoxy(25,18);Write('En el punto final derecho: ');
     Gotoxy(21,24);
     Write('Presione cualquier tecla para continuar');
     repeat
       Gotoxy(54,15);ReadIn(DerivFI);
       RevisaES;
     until not ErrES;
     repeat
       Gotoxy(54,18);ReadIn(DerivFD);
       RevisaES;
     until not ErrES;
  Gotoxy(60,24);Cont := readkey;
  Encabezado;
  Gotoxy(11,8);Write('Entrada de puntos a Interpolar');
  Obten1VecDeTeclado(NumPtsInter, XInter);
  ObtenArchSalida(ArchSal);
end;

procedure Resultados(NumPunts : integer;
                     var DatoX, DatoY : vector;
                     DerivFI, DerivFD  : real;
                     var Coef0, Coef1, Coef2, Coef3 : vector;
                     NumPtsInter : integer;
                     var XInter, YInter : vector; Error : byte);
var Indice : integer;

procedure Titulo;
begin
  ClrScr;
  Writeln(ArchSal);
  Writeln(ArchSal);
  Gotoxy(1,2);
  Writeln(ArchSal,' ':24,'INTERPOLADOR CUBICO POR SEGMENTOS');
  Writeln(ArchSal);
  Gotoxy(21,24);Write('Presione cualquier tecla para continuar');
end;
```

```
begin
  Writeln(ArchSal);
  Writeln(ArchSal);
  Titulo;
  Gotoxy(1,5);Writeln(ArchSal, ' ':37,'DATOS');
  Writeln(ArchSal);
  Writeln(ArchSal);
  Gotoxy(1,7);
  Writeln(ArchSal,' ':18,'Dato :          X                    Y');
  for Indice := 1 to NumPunts do
    begin
      Gotoxy(1,8+Indice);Writeln(ArchSal,' ':17,' ',Indice:3,' :    ',
                          DatoX[Indice]:15:10,'    ',
                          DatoY[Indice] : 15 : 10);
    end;
  Writeln(ArchSal);
  Gotoxy(1,10+Indice);Writeln(ArchSal,' ':15,'Derivada en X = (',
                          DatoX[1],') : ',DerivF1);
  Gotoxy(1,12+Indice);Writeln(ArchSal,' ':15,'Derivada en X = (',
                          DatoX[NumPunts],') :  ',DerivFD);
  Writeln(ArchSal);
  if Error <> 1 then
    DespliegaErr;
  {if Pantalla = 1 then Gotoxy(60,24);Cont := readkey;}
  case Error of
    0 : begin
          Titulo;
          Writeln(ArchSal);
          Gotoxy(1,8);Writeln(ArchSal, 'Segmento:',' ':6,
                          'Coef0', ' ':13,'Coef1', ' ':14,
                          'Coef2', ' ':13,'Coef3');
          for Indice := 1 to NumPunts-1 do
            begin
              Gotoxy(4,9+Indice);
              Writeln(ArchSal, Indice : 3,
                    ':', ' ', Coef0[Indice]:15:10,' ':3,
                             Coef1[Indice]:15:10, ' ':3,
                             Coef2[Indice]:15:10, ' ':3,
                             Coef3[Indice]:15:10);
            end;
          Writeln(ArchSal);
          {if Pantalla = 1 then Gotoxy(60,24);Cont := readkey;}
          Titulo;
          Writeln(ArchSal);
          Gotoxy(1,4);Writeln(ArchSal,' ':35,'RESULTADOS');
          Writeln(ArchSal);
          Gotoxy(1,7);Writeln(ArchSal,' ':31,'Puntos interpolados');
          Writeln(ArchSal);
          Gotoxy(1,10);
          Writeln(ArchSal,' ':23, 'Puntos:', ' ':13,'X', ' ':18,'Y');
```

```
        for Indice := 1 to NumPtsInter do
          begin
            Gotoxy(1,11+Indice);
            Writeln(ArchSal,' ':22,' ',Indice:3, ':    ',
                              XInter[Indice] : 15 : 10, '    ',
                              YInter[Indice] : 15 : 10);
          end;
        {if Pantalla = 1 then Gotoxy(60,24);Cont := readkey;}
      end;
    1 : begin
        Titulo;
        Gotoxy(25,15);
        Writeln(ArchSal, 'Los puntos X deben ser unicos.');
        Gotoxy(60,24);Cont := readkey;
      end;
    2 : begin
        Titulo;
        Gotoxy(28,15);
        Writeln(ArchSal,'Los puntos X deben llevar');
        Gotoxy(28,17);
        Writeln(ArchSal,'un incremento secuencial ');
        Gotoxy(60,24);Cont := readkey;
      end;
    3 : begin
        Titulo;
        Gotoxy(18,15);
        Writeln(ArchSal, 'Estos deben ser los dos ilitimos puntos.');
        Gotoxy(60,24);Cont := readkey;
      end;
  end;
end;


begin
  Inicializa(Coef0, Coef1, Coef2, Coef3,
          DatoX, DatoY, DerivF1, DerivFD,
          NumPunts, NumPtsInter, XInter, YInter, Error);
  Captura(NumPunts, DatoX, DatoY, DerivF1, DerivFD,
          NumPtsInter, XInter);
  TrazadorSeg(NumPunts, DatoX, DatoY, DerivF1, DerivFD,
          NumPtsInter, XInter, Coef0, Coef1, Coef2,
          Coef3, YInter, Error);
  Resultados(NumPunts, DatoX, DatoY, DerivF1, DerivFD,
          Coef0, Coef1, Coef2, Coef3, NumPtsInter,
          XInter, YInter, Error);
  Close(ArchSal);
end.
```

INTERPOLACIÓN EN DOS DIMENSIONES

En la presente técnica consideraremos la aproximación de la función f = f(x,y). Partiendo de un tabla bidimensional de valores de f (x,y) de tamaño m*n, para todas las combinaciones posibles de los m datos de x y los n datos de y, donde el renglón corresponderá a x_i y la columna a y_i. Por tanto para cada pareja arbitraria x e y el problema trata de una interpolación dentro de una tabla para localizar una aproximación de f(x,y).

El procedimiento se realiza en tres pasos que consisten en:

A. Interpolación lineal para x a través de $f(x_i,y_j)$ y $f(x_{i+1},y_j)$.

B. Interpolación lineal para y entre $f(x_i,y_{j+1})$ y $f(x_{i+1},y_{j+1})$.

C. Interpolación lineal entre los valores obtenidos en el paso A y el paso B. obteniendo de esta manera la aproximación de f(x,y).

Algoritmo

Para encontrar un valor interpolado dada una tabla de datos en dos dimensiones.

Entrada nx, ny número de datos tanto para x como para y,
x_i, y_i $f(x_i,y_i)$ valores de los datos,
x, y valores a interpolar.

Salida x, y, f(x,y) valor interpolado, Datos fuera de rango.

Paso 1 para i = 1, ..., n-1 hacer 2

Paso 2 Si ($x \geq x_{i-1}$ y $x \leq x_i$ y ($x \geq x_i$ y $x \leq x_{i+1}$) entonces Paso 3
sino Paso 10

Paso 3 II = i

Paso 4 para i = 1, ..., n-1 hacer 5

Paso 5 Si ($y \geq y_{i-1}$ y $y \leq y_i$ y ($y \geq y_i$ y $y \leq y_{i+1}$) entonces Paso 6
sino Paso 10

Paso 6 JJ = i

Paso 7

$$f(x, y_{JJ}) = f(x_{II}, y_{JJ}) + \frac{f(x_{II-1}, y_{JJ}) - f(x_{II}, y_{JJ})}{x_{II-1} - x_{II}} (x - x_{II})$$

Paso 8

$$f(x, y_{JJ-1}) = f(x_{II}, y_{JJ-1}) + \frac{f(x_{II-1}, y_{JJ-1}) - f(x_{II}, y_{JJ-1})}{x_{II-1} - x_{II}} (x - x_{II})$$

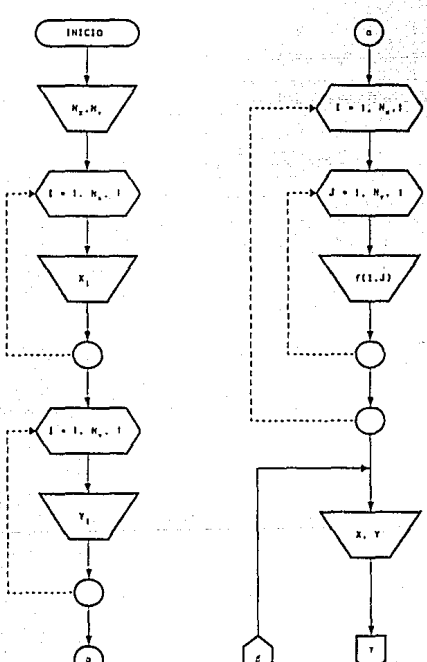
Paso 9

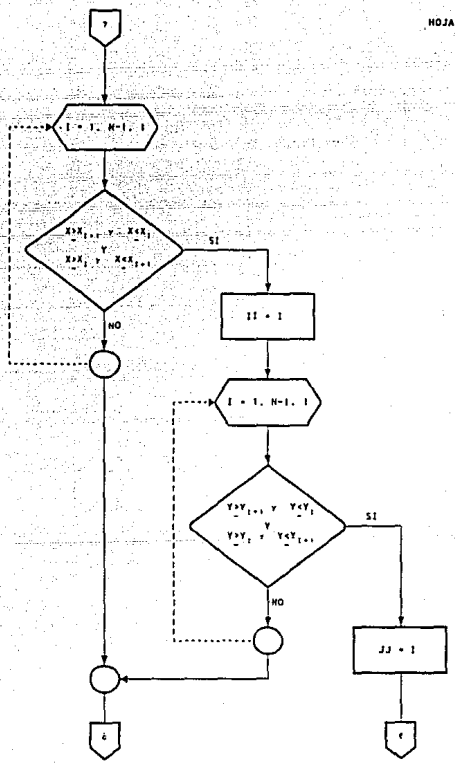
$$f(x, y) = f(x, y_{JJ-1}) - \frac{f(x, y_{JJ-1}) - f(x, y_{JJ})}{x_{II-1} - x_{II}} (y - y_{II})$$

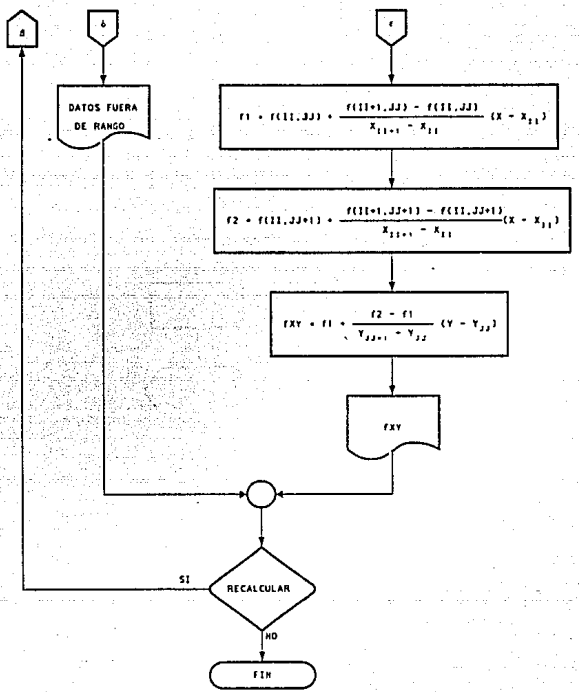
Paso 8 Salida (x, y, f(x,y)), Terminar.

Paso 9 Salida (Datos fuera de rango), Terminar.

INTERPOLACION EN DOS DIMENSIONES




7
I = 1, N-1, 1
X>X_{I+1} Y X<X_I
Y
X>X_I Y X<X_{I+1}
SI
NO
II = I
I = 1, N-1, 1
Y>Y_{I+1} Y Y<Y_I
Y
Y>Y_I Y Y<Y_{I+1}
SI
NO
JJ = I
6
8


a
b
c
DATOS FUERA DE RANGO
$f1 = f(II,JJ) + \frac{f(II+1,JJ) - f(II,JJ)}{X_{II+1} - X_{II}} (X - X_{II})$
$f2 = f(II,JJ+1) + \frac{f(II+1,JJ+1) - f(II,JJ+1)}{X_{II+1} - X_{II}} (X - X_{II})$
$FXY = f1 + \frac{f2 - f1}{Y_{JJ+1} - Y_{JJ}} (Y - Y_{JJ})$
FXY
SI
RECALCULAR
NO
FIN

```
program Interpolacion_en_dos_dimensiones;
uses Crt;
var i, j, k, ii, jj, nc, nr, nx, z, mensaje : integer;
   xi, yi, f1, f2, fxy : real;
   x : array [1..20] of real;
   y : array [1..40] of real;
   f : array [1..20,1..40] of real;
   c : char;

procedure Inicializa;
begin
  i   := 0;
  j   := 0;
  k   := 0;
  ii  := 0;
  jj  := 0;
  nc  := 0;
  nr  := 0;
  nx  := 0;
  z   := 0;
  xi  := 0;
  yi  := 0;
  f1  := 0;
  f2  := 0;
  fxy := 0;
  mensaje := 0;
  for i := 1 to 20 do
    begin
      x[i] := 0;
      for j := 1 to 40 do
        begin
          y[j] := 0;
          f[i,j] := 0;
        end
    end;
  c := char(0);
end;

procedure Captura;
begin
  clrscr;
  gotoxy(24,5);write('INTERPOLACION EN DOS DIMENSIONES');
  gotoxy(25,8);write('Suministre los siguientes datos');
  gotoxy(27,13);write('Número de renglones: r = ');
  gotoxy(27,16);write('Número de columnas:  c = ');
  gotoxy(21,24);write('Presione cualquier tecla para continuar');
  gotoxy(52,13);read(nr);
  gotoxy(52,16);read(nc);
  gotoxy(60,24);c := readkey;
  clrscr;
```

```
  gotoxy(24,2);write('INTERPOLACION EN DOS DIMENSIONES');
  gotoxy(25,5);write('Suministre los siguientes datos');
  gotoxy(25,5);write('VARIABLE                FUNCION');
  gotoxy(10,6);write('INDEPENDIENTE  DEPENDIENTE  f (x,y)');
  for i := 1 to nr do
    begin
      gotoxy(10,7+i);write('x[',i,'] = ');
      gotoxy(18,7+i);read(x[i]);
    end;
  for j := 1 to nc do
    begin
      gotoxy(32,7+j);write('y[',j,'] = ');
      gotoxy(39,7+j);read(y[j]);
    end;
  for i := 1 to nr do
    for j := 1 to nc do
      begin
        gotoxy(52,7+j);write('f[',i,',',j,'] = ');
        gotoxy(62,7+j);read(f[i,j]);
      end;
  clrscr;
  gotoxy(25,5);write('INTERPOLACION EN DOS DIMENSIONES');
  gotoxy(29,8);write('Suministre lo siguiente');
  gotoxy(31,11);write('Datos a interpolar');
  gotoxy(17,13);write('x = ');
  gotoxy(43,13);write('y = ');
  gotoxy(21,24);write('Presione cualquier tecla para continuar');
  gotoxy(21,13);read(xi);
  gotoxy(47,13);read(yi);
end;

procedure Proceso;
begin
  f1 := f[ii,jj] + ( (f[ii+1,jj] - f[ii,jj])
       / (x[ii+1] - x[ii]) ) * (xi-x[ii]);
  f2 := f[ii,jj+1] + ( (f[ii+1,jj+1] - f[ii,jj+1])
       / (x[ii+1]-x[ii]) ) * (xi-x[ii]);
  fxy := f1 + ( (f2 - f1) / (y[jj+1] - y[jj]) ) * (yi-y[jj]);
end;

procedure Interpolacion2v;
begin
 repeat
  for i := 1 to nr-1 do
  begin
  if ((xi >= x[i]) and (xi <= x[i+1])) or
     ((xi <= x[i]) and (xi >= x[i+1])) then
       begin
         ii := i;
```

```
    for j := 1 to nc-1 do
      if ((yi >= y[j]) and (yi <= y[j+1])) or
        ((yi <= y[j]) and (yi >= y[j+1])) then
          begin
            JJ := j;
            Proceso;
            mensaje := 1;
          end
    end;
  end;
 until mensaje <> 0;
end;

procedure Resultado;
begin
  if mensaje = 1 then
    begin
      gotoxy(1,17);clreol;
      gotoxy(34,17);write('Interpolación');
      gotoxy(12,19);
      write('x=',xi:5:8,'y=',yi:5:8,'f(x,y)=',fxy:7:8);
    end
   else
    begin
      gotoxy(1,17);clreol;
      gotoxy(30,17);write('Dato a interpolar fuera de rango');
    end;
  gotoxy(60,24);c := readkey;
end;

begin
  Inicializa;
  Captura;
  Interpolacion2v;
  Resultado;
end.
```

CAPITULO III

ECUACIONES ALGEBRAICAS NO LINEALES

ECUACION ALGEBRAICA NO LINEAL

Este tipo de ecuaciones incluye una gran diversidad de ecuaciones que surgen del análisis de procesos Físicos y/o Químicos cuya estructura es muy diversa, desde formas polinomiales con potencias de la incógnita diferentes de uno hasta la inclusión de funciones logarítmicas, exponenciales, trigonométricas o recíprocas las cuales normalmente no pueden resolverse analíticamente y es necesario el uso de algún método numérico.

La técnicas numéricas para resolver una ecuación algebraica no lineal se basan en la búsqueda de una raíz de una función, es decir, la ecuación que está representada por alguna estructura de la forma

$$g(x) = c$$

siendo c una constante, se modifica para presentarla como

$$f(x) = g(x) - c = 0$$

de tal manera que el objetivo sea buscar un valor de x para el cual f(x) o g(x)-c sea cero, el cual, por definición es una raíz de la función f(x).

Con esta reestructuración se pretende utilizar algoritmos con los que se estime el valor de una raíz de la función.

En procesos Físicos y/o Químicos, la raíces que se buscan por lo general son raíces reales y en este capítulo presentaremos las técnicas fundamentales para la evaluación de éstas.

SERIE DE TAYLOR

Siendo f una función y n un entero positivo tal que la derivada $f^{(n+1)}(x)$ exista para cada x en un intervalo I. Si a y b son valores distintos en I, entonces hay un número z entre a y b tal que

$$f(b) = f(a) + \frac{f'(a)}{1!}(b-a) + \frac{f''(a)}{2!}(b-a)^2 \ldots \frac{f^{(n)}(a)}{n!}(b-a)^n + \frac{f^{(n+1)}(z)}{(n+1)!}(b-a)^{(n+1)}$$

como se ve esta serie es infinita, para obtener una expresión finita y evaluable utilizaremos el término residuo, R_n, es cual está dado por la fórmula

$$R_n = \frac{f^{(n+1)}(z)}{(n+1)!}(b-a)^{n+1}$$

R_n depende de a, b y n.

Este teorema es la base para el desarrollo de algoritmos orientados a la búsqueda de raíces y que utilizan derivadas como parte de la estrategia de localización. Un método representativo de esta estrategia es el Newton-Raphson.

MÉTODO DE NEWTON-RAPHSON

El presente método es uno de los más ampliamente utilizados para calcular una raíz de la función f(x), un valor de x tal que f(x) sea igual a cero, con lo cual estaremos resolviendo la ecuación f(x) = 0.

1.- Expanda linealmente mediante serie de taylor la función f(x), a partir de un punto base x_o.

2.- Aplique la condición de raíz, x es raíz si f(x) es igual a cero.

Expansión :

$$f(x) = f(x_o) + f'(x_o) * (x - x_o)$$

Condición x es raíz, f(x) = 0

$$0 = f(x_o) + f'(x_o) * (x - x_o)$$

rearreglando:

$$x = x_o - \frac{f(x_o)}{f'(x_o)}$$

Este algoritmo no dá directamente la solución porque f(x) es no lineal y realmente no corresponde a la línea a la que se está aproximando por expansión.

La expansión está suministrando una estructura lineal que supuestamente representa a f(x) en torno a x_o, esto podría ser cierto si x estuviese muy cerca del valor de la raíz, de tal manera que la diferencia $(x-x_o)$ tendiese a cero y en consecuencia, los demás términos de la serie que están incluidos en el error de truncación contendrían términos $(x-x_o)^2$, $(x-x_o)^3$, etc. serían prácticamente cero y se estaría garantizando que f(x) se comporte como línea recta en el intervalo $[x,x_o]$.

Dado que, por lo general, el error de truncación no es cero porque x_o no está próxima a un valor de la raíz, el método tan sólo genera una estrategia de predicción-corrección que debe aplicarse iterativamente.

La representación gráfica del método de Newton-Raphson esta en la siguiente página.

La expansión lineal indica que f(x) está representada por una línea recta que pasa por el punto $(x_o,f(x_o))$, cuya pendiente es $f'(x_o)$.

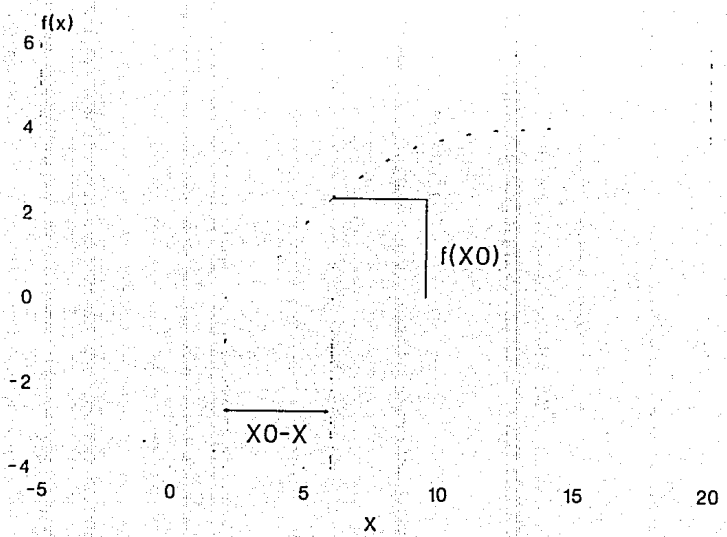
$f(x) = f(x_o) + f'(x_o) * (x-x_o)$, corresponde a la línea:

$$f(x) = f(x_o) - x_o * f'(x_o) + f'(x_o) * x,$$

$$f(x) = a_o + mx$$

Donde m es la pendiente: $m = f'(x_o)$ y a_o es la ordenada al origen : $a_o = f(x_o) - x_o * f'(x_o)$.

METODO DE NEWTON



Sobre esta línea deberá buscarse el valor de x para el cual f(x) es cero, dado que la línea no representa correctamente a la función debido al error de truncación, este valor de x tan sólo es aceptado como un nuevo punto base x_o, hasta que se cumpla que el valor de $f(x_o)$ esté dentro del margen de aceptación.

Algoritmo:

Encontrar una solución de $f(x) = 0$ dada la aproximación inicial x_o:

Entrada: x_o aproximación inicial o punto base de la expansión. p la tolerancia.

Salida: x_o solución aproximada, mensaje no converge.

Paso 1 Para i = 1, ... n realizar pasos 2,3,4

Paso 2 Si $| f(x_o) | < p$ entonces Salida (x_o) Termina

Paso 3 Si $f'(x_o) = 0$ entonces $x_o = x_o + p$

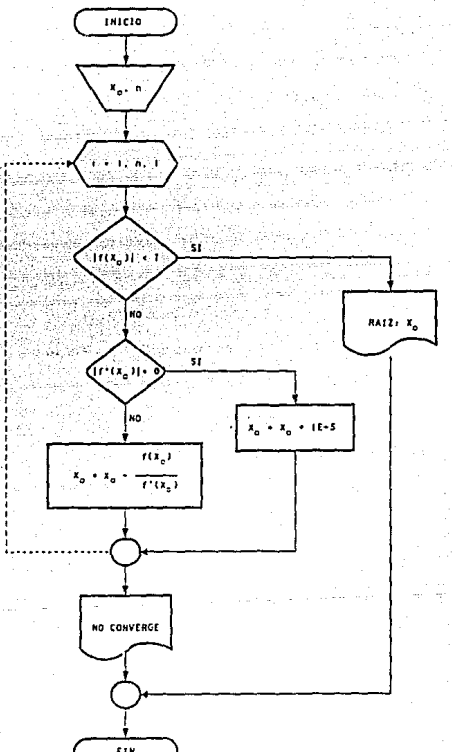
Paso 4 $x_o = x_o - f(x_o)/f'(x_o)$

Paso 5 Salida (No converge)

Termina.

siendo p la tolerancia o margen de error, $f(x_o)$, $f'(x_o)$ la evaluación de la función y la derivada en el punto correspondiente.

METODO DE NEWTON-RAPHSON



```
program Metodo_Newton_Raphson;
uses Crt;

var I : Integer;
   X, t: real;
   C : Char;

function F(X : real) : real;
begin
  F := X*X + 4*X - 2;
end;

function DF(X : real) : real;
begin
  DF := 2*X + 4;
end;

procedure Inicializa;
begin
  i := 0;
  t := 0;
  x := 0;
  c := char(0);
end;

procedure Capturat(t, X : real);
begin
  clrscr;
  gotoxy(30,5);write('METODO NEWTON-RAPHSON');
  gotoxy(23,8);writeln('Función a evaluar F(x) = x^2 +4x -2');
  gotoxy(24,10);writeln'Suministre los siguientes datos :');
  gotoxy(30,14);write('Valor inicial (Xo) = ');
  gotoxy(30,17);write('Tolerancia    (t)  = ');
  gotoxy(20,24);write('Presione cualquier tecla para continuar:');
  gotoxy(51,14);read(X);
  gotoxy(51,17);read(t);
  gotoxy(60,24);c := readkey;
end;

procedure Resultado(i : Integer; X: real);
begin
  clrscr;
  gotoxy(30,5);write('METODO NEWTON-RAPHSON');
  gotoxy(23,8);writeln('Función evaluada F(x) = x^2 +4x -2');
  gotoxy(18,12);write('La raíz obtenida es  x = ',X);
  gotoxy(31,14);writeln('con ',i,' iteraciones');
  gotoxy(20,24);writeln'Presione cualquier tecla para continuar:');
  gotoxy(60,24);c := readkey;
  exit;
end;
```

```
procedure NoConverge(i : Integer; X : real);
var N : integer;
begin
 clrscr;
 gotoxy(30,5);write('METODO NEWTON-RAPHSON');
 gotoxy(24,8);write('Función evaluada F(x) = cosx - x');
 gotoxy(15,12);write('El método no converge en ',i,' iteraciones dando x = ',x);
 gotoxy(10,14);write('para la tolerancia requerida, para solucionar');
 gotoxy(54,14);write('esto aumente el');
 gotoxy(10,16);write('número de iteraciones o tomar');
 gotoxy(39,16);write(' una raiz con menor tolerancia');
 gotoxy(21,24);write('Presione cualquier tecla para continuar');
 gotoxy(60,24);c := readkey;
end;

procedure Newton(i : Integer; t, X: real);
begin
   while ABS(F(X)) > t do
     begin
       i := succ(i);
       if DF(X) = 0 then
                  X := X + t;
       X := X - F(X)/DF(X);
       if i >= 100 then
                 begin
                   NoConverge(i,x);
                   exit;
                 end;
     end;
  Resultado(i, X);
end;

Begin
  Inicializa;
  Capturar(t, X);
  Newton(i, t, X);
end.
```

EXPANSION MEDIANTE: POLINOMIO FUNDAMENTAL DE NEWTON

Polinomio Fundamental de Newton :

Si f(x) es una función continua en el intervalo finito [a,b], entonces para cada $\epsilon > 0$, habra un polinomio P(x) tal que $| f(x) - P(x) | < \epsilon$ para toda x en el intervalo dado.

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

$$P_n(x) = f[x_0] + (x-x_0) f[x_1, x_0] + (x-x_0)(x-x_1) f[x_2, x_1, x_0] + \ldots + (x-x_0)(x-x_1)(x-x_2)\ldots f[x_n, x_{n-1}, \ldots, x_0]$$

$$R_n(x) = (x-x_0)(x-x_1)\ldots(x-x_{n-1})(x-x_n) f[x_n, x_{n-1}, \ldots, x_0]$$

donde

$$f[x_i] = f(x_i)$$

$$f[x_1, x_i] = \frac{f[x_i] - f[x_0]}{x_i - x_0}$$

llamadas diferencias divididas.

Esta expansión ha generado una serie de algoritmos, a continuación se presentan los más importantes.

MÉTODO DE LA SECANTE

El método de la secante es otra de las técnicas para calcular un valor de x tal que f(x) sea cero. Si se expande linealmente a f(x) mediante el polinomio fundamental de Newton, se obtiene la siguiente ecuación :

$$f(x) = f(x_0) + f[x_0,x_1] * (x - x_0)$$

Esta ecuación representa a la función como si fuese la línea recta que pasa por los puntos $(x_0,f(x_0))$, $(x_1,f(x_1))$, es decir es una secante de la curva real, la cual pasa por los puntos mencionados.

Dado que estamos buscando un valor de x tal que f(x) sea cero, consideramos que esta expansión es representativa de la función hasta el punto en que ésta vale cero, por lo tanto, de la ecuación anterior se obtiene :

$$0 = f(x_0) + f[x_0,x_1] * (x - x_0)$$

la cual se rearregla para obtener:

$$x = \frac{x_0 * f(x_1) - x_1 * f(x_0)}{f(x_1) - f(x_0)}$$

Dado que la línea no corresponde exactamente a la función, este nuevo valor de x podrá no ser la raíz buscada y tan sólo es una aproximación, entonces este nuevo valor se considera como la siguiente aproximación. En la página 74 está la representación gráfica de este método.

Algoritmo:

Encontrar una solución de $f(x) = 0$, dadas las aproximaciones iniciales x_0, x_1 (valores supuestos):

Entrada: x_0, x_1 aproximaciones iniciales y t la tolerancia.

Salida: x solución aproximada, o mensaje no converge.

Paso 1 para i = 1, ..., n realizar Pasos 2, 3 y 4.

Paso 2

$$x = \frac{x_0 * f(x_1) - x_1 * f(x_0)}{f(x_1) - f(x_0)}$$

Paso 3 Si $| f(x) | < t$ entonces Salida (x). Termina.

Paso 4 $x_0 = x_1$; $x_1 = x$; ir a Paso 2

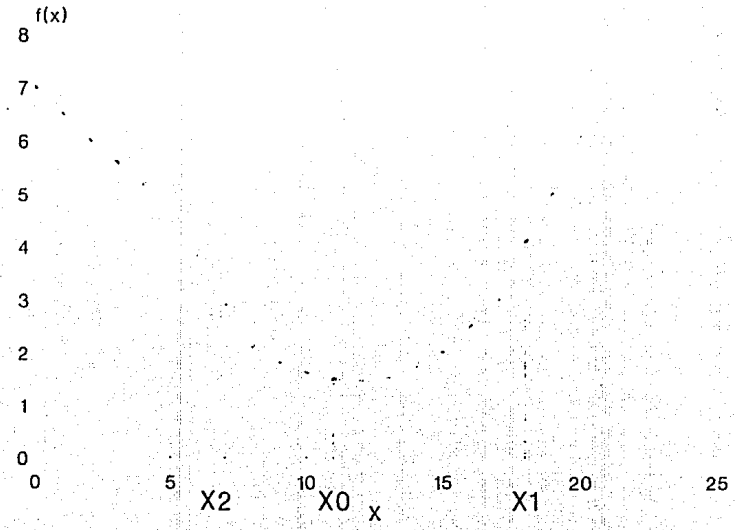
Paso 5 Salida (No converge)

Termina.

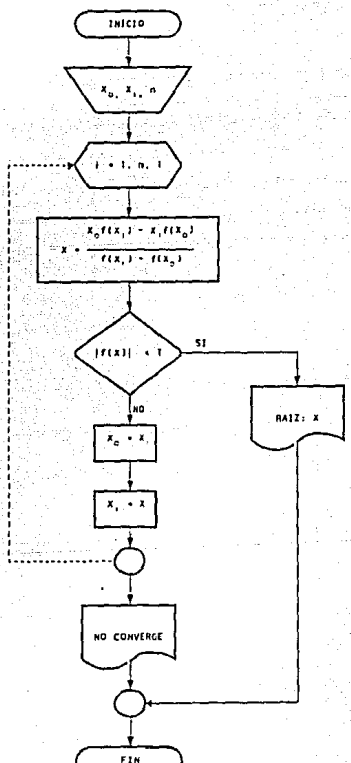
siendo t la tolerancia, $f(x)$, $f(x_1)$, $f(x_0)$ la evaluación de la función en los puntos correspondientes.

Este método converge si los valores x_0 y x_1 están muy cerca de la raíz, aunque en la mayoría de los casos, la falta de información sobre un valor aproximado de la raíz y la complejidad del modelo hace divergente al método.

METODO DE LA SECANTE



METODO DE LA SECANTE



```
Program Regla_de_la_Secante;
Uses crt;
var X, X0, X1, t : real;
    i,j : Integer;
    c : Char;

function F(X : real) : real;
begin
  F := X*X + 4*X - 2;
end;

procedure Inicializa;
begin
  i := 0;
  j := 0;
  t := 0;
  X0 := 0;
  X1 := 0;
  c := char(0);
end;

procedure Captura(var t,X0,X1: real);
begin
  clrscr;
  gotoxy(30,5);write('METODO DE LA SECANTE');
  gotoxy(23,8);write('Función a evaluar F(x) = x^2 +4x -2');
  gotoxy(24,12);write('Suministre los valores iniciales:');
  gotoxy(25,15);write('X[0] = ');
  gotoxy(47,15);write('X[1] = ');
  gotoxy(33,17);write('tolerancia = ');
  gotoxy(20,24);write('Presione cualquier tecla para continuar:');
  gotoxy(32,15);read(X0);
  gotoxy(54,15);read(X1);
  gotoxy(45,17);read(t);
  gotoxy(60,24);c := readkey;
end;

procedure Resultado(i : Integer; X: real);
begin
  clrscr;
  gotoxy(30,5);write('METODO DE LA SECANTE');
  gotoxy(23,8);write('Función evaluada F(x) = x^2 +4x -2');
  gotoxy(18,12);write('La raíz obtenida es  x = ',X);
  gotoxy(31,14);write('con ',j,' iteraciones');
  gotoxy(20,24);write('Presione cualquier tecla para continuar:');
  gotoxy(60,24);c := readkey;
  exit;
end;
```

```
procedure NoConverge(i : Integer; X : real);
var N : integer;
begin
 clrscr;
 gotoxy(30,5);writeln('METODO DE LA SECANTE');
 gotoxy(24,8);writeln('Función evaluada F(X) = cosx - x');
 gotoxy(15,12);writeln('El método no converge en ',i,' iteraciones');
 gotoxy(10,14);writeln('para la tolerancia requerida, para solucionar');
 gotoxy(54,14);writeln(' esto aumente el');
 gotoxy(10,16);writeln('número de iteraciones a tomar');
 gotoxy(39,16);writeln(' una raiz con mayor tolerancia');
 gotoxy(21,24);writeln('Presione cualquier tecla para continuar');
 gotoxy(60,24);c := readkey;
end;

procedure Secante(t:integer;t,X0,X1:real; var X:real;
                  var j:integer);
begin
  repeat
    i := i + 1;
    X := (X0*F(X1) - X1*F(X0))/(F(X1) - F(X0));
    if ABS(F(X)) > t then
       begin
         X0 := X1;
         X1 := X;
       end;
    if i >= 100 then
              begin
                NoConverge(i,x);
                halt;
              end;
    j:=i;
  until ABS(F(X)) < t;
end;

begin
  Inicializa;
  Captura(t,X0,X1);
  Secante(i,t,X0,X1,X,j);
  Resultado(j,X);
end.
```

MÉTODO REGULA-FALSI (POSICIóN FALSA)

Una mejora del anterior método está representado por el método de Regula Falsi, el cual como estrategia de aproximación utiliza el método de la secante, modificándolo en el criterio de reemplazos de x_0 y x_1.

Una vez evaluado el valor de x, se verifica en cuál de los intervalos $[x_0,x]$ y $[x,x_1]$ está contenida una raíz de f(x). Esta verificación se lleva a cabo aplicando el Teorema de existencia de raíz de una función continua dentro de un cierto intervalo:

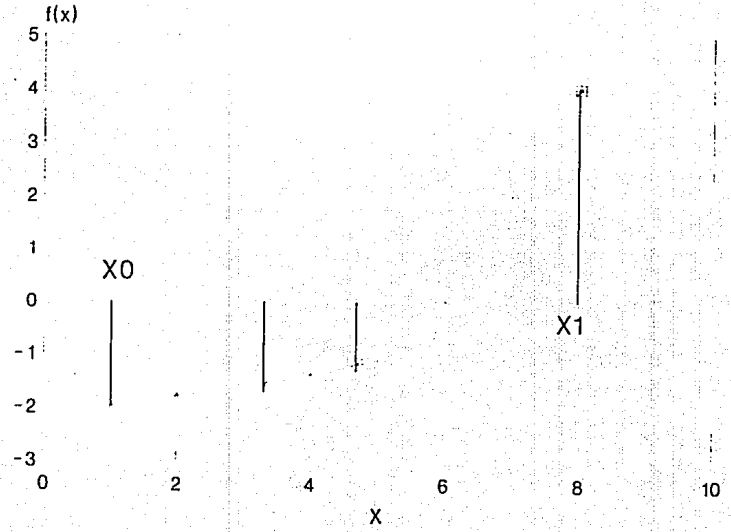
Teorema de existencia de raíz:

Sea x una raíz de la función f(x) y a un valor de x, además de que la función sea continua. Si f(a) = 0 se dice que el valor a es una raíz de la función.

La aplicación de este teorema nos conduce a verificar si el producto de $f(x)*f(x_0)$ es menor que cero, si este es el caso entonces en el intervalo $[x_0,x]$ existe un punto tal que f(x) es cero y en consecuencia x_0 es un punto adecuado para conservarlo, por lo que se reemplaza el valor de x_1, si no es éste el caso entonces se reemplaza x_0 por x.

La representación gráfica del método se encuentra en la siguiente página.

REGULA - FALSI



Algoritmo:

Encontrar una solución de $f(x) = 0$, dadas las aproximaciones iniciales x_0, x_1 (supuestas):

Entrada: x_0, x_1 aproximaciones iniciales, la tolerancia: t.

Salida: x solución aproximada, no converge.

Paso 1 para i = 1, ..., n hacer

Paso 2

$$x = \frac{x_0 * f(x_1) - x_1 * f(x_0)}{f(x_1) - f(x_0)}$$

Paso 3 Si $| f(x) | < t$ entonces Salida (x), Terminar.

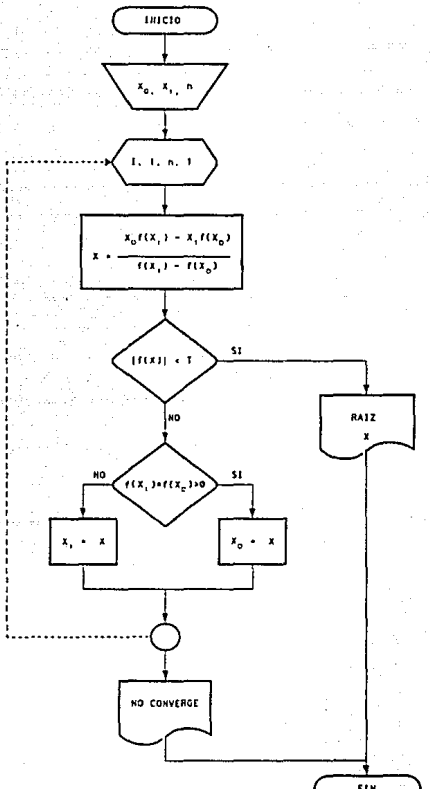
Paso 4 Si $f(x_0) * f(x_1) < 0$ entonces $x_0 = x$, e ir a Paso 2

sino $x_1 = x$, e ir a Paso 2

Paso 5 Salida (No converge)

Terminar.

METODO DE REGULA-FALSI



```
Program Metodo_de_la_Regula_falsi;
uses crt;

var X0, X1, X, T :real;
    i : integer;
    c : char;
    Solucion : boolean;

function F(X : real) : real;
begin
   F:= X*X + 4*X - 2;
end;

procedure Inicializa;
begin
  i := 0;
  t := 0;
  X := 0;
  X0 := 0;
  X1 := 0;
  c := char(0);
  Solucion := False;
end;

procedure Captura;
begin
  clrscr;
  gotoxy(32,5);write('METODO REGULA FALSI').
  gotoxy(23,8);write('Función a evaluar F(x) = x^2 +4x -2');
  gotoxy(24,10);write('Suministre los siguientes datos :');
  gotoxy(30,14);write('Valor inicial (Xo) = ');
  gotoxy(30,16);write('Valor inicial (X1) = ');
  gotoxy(30,18);write('Tolerancia    (t)  = ');
  gotoxy(20,24);write('Presione cualquier tecla para continuar:');
  gotoxy(51,14);read(X0);
  gotoxy(51,16);read(X1);
  gotoxy(51,18);read(T);
  gotoxy(60,24);c := readkey;
end;

procedure Regula;
begin
  repeat
    i := succ(i);
    X := (X0*F(X1) - X1*F(X0))/(F(X1) - F(X0));
    if ABS(F(X)) < T then  Solucion := true;
    if F(X)*F(X0) > 0 then X0 := X
                      else X1 := X;
  until Solucion = true;
end;
```

```
procedure Resultado;
begin
  clrscr;
  gotoxy(32,5);write('METODO REGULA FALSI');
  gotoxy(23,8);write('Función evaluada F(x) = x^2 +4x -2');
  gotoxy(18,12);write('La raíz obtenida es  x = ',X);
  gotoxy(31,14);write('con ',i,' iteraciones');
  gotoxy(20,24);write('Presione cualquier tecla para continuar:');
  gotoxy(60,24);c := readkey;
end;

begin
  Inicializa;
  Captura;
  Regula;
  Resultado;
end.
```

MÉTODO DE WENGSTEIN

Este método trata de evitar el error de redondeo y entonces modifica la definición de la función tratando de que posea valores altos.

Sea f(x) la función a la cual se desea evaluar una raíz, es decir, encontrar un valor de x tal que f(x) sea cero.

Defina g(x) = f(x) + x. Si x es raíz, g(x) será igual a x dado que f(x) es cero. Expandiendo linealmente a g(x) con el polinomio fundamental de Newton:

$$g(x) = g(x_o) + g[x_o,x_1] * (x - x_o)$$

representa una línea que pasa por los puntos $(x_o,g(x_o))$, $(x_1,g(x_1))$. Sobre esta línea habrá un punto tal que g(x) sea igual a x y será éste el valor propuesto para la raíz.

Aplicando la condición: Si x es raíz g(x) = x se obtiene:

$$x = g(x_o) + g[x_o,x_1] * (x - x_o)$$

rearreglando esta ecuación :

$$x = \frac{x_o * g(x_1) - x_1 * g(x_o)}{g(x_1) - g(x_o) + x_o - x_1}$$

Este valor de x será la nueva aproximación para la raíz. En la siguiente página se muestra la representación del método.

Se traza la línea auxiliar $g(x) = x$ (diagonal principal), dado que cualquier solución posible estará sobre esta línea y se traza la curva $g(x)$ que corresponde a $f(x) + x$. En la figura se muestran los dos puntos iniciales propuestos y cómo genera y se localiza un nuevo punto en la intersección de la línea de expansión y la línea auxiliar.

Algoritmo:

Para encontrar una solución de $g(x) = f(x) + x = 0$, dadas las aproximaciones iniciales x_0 y x_1 (supuestas).

Entrada: x_0, x_1 aproximaciones iniciales , la tolerancia t.

Salida: x_0 solución aproximada, no converge.

Paso 1 para i = 1, ..., n hacer pasos 2, 3

Paso 2

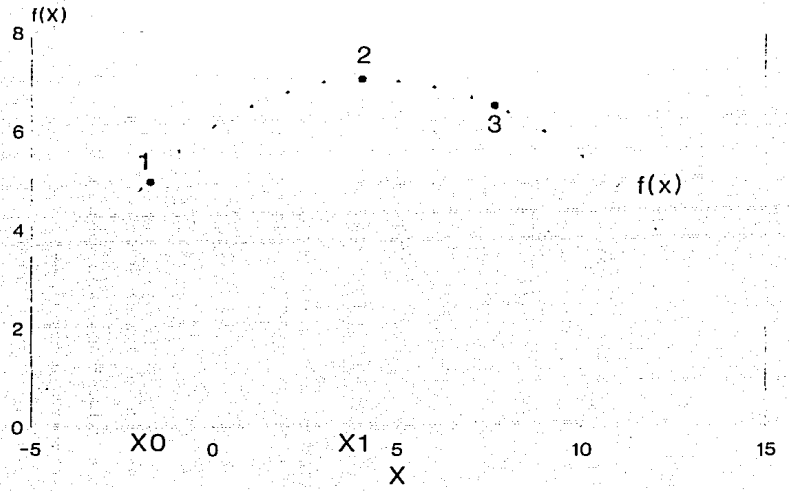
$$x = \frac{x_0 * g(x_1) - x_1 * g(x_0)}{g(x_1) - g(x_0) + x_0 - x_1}$$

Paso 3 Si $| f(x_0) | < t$ entonces Salida (x_0). Terminar. **Paso 4** $x_0 = x_1$; $x_1 = x$, e ir al Paso 2

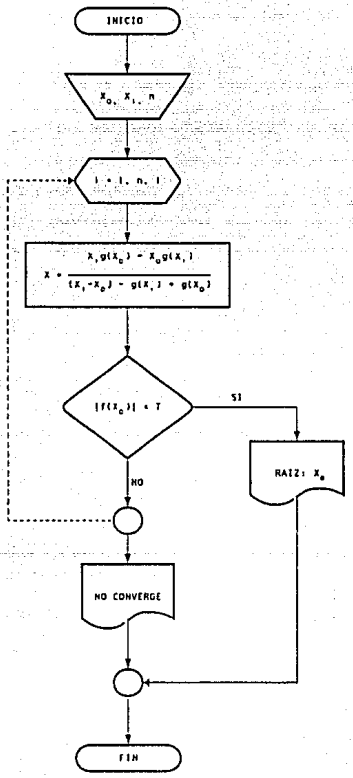
Paso 5 Salida (No converge)

Terminar.

METODO DE WENGSTEIN



METODO DE WENGSTEIN



```
Program  Metodo_de_Wengstein;
uses Crt;

var X0, X1, t : real;
    i, j : Integer;
    c : char;
    mensaje : byte;

function G(x : real) : real;
begin
  G := Exp(-X);
end;

function F(x : real) : real;
begin
  F := Exp(-X) - X;
end;

procedure Inicializa;
begin
  X0 := 0;
  X1 := 0;
  i  := 0;
  j  := 1; t := 0;
  mensaje := 0;
  c  := char(0);
end;

procedure Captura;
begin
  clrscr;
  gotoxy(32,3);write('METODO  WENGSTEIN');
  gotoxy(26,6);write('Función evaluada: Exp(-X) - X');
  gotoxy(25,9);write('Suministre los siguientes datos');
  gotoxy(32,12);write('Valor de  X0 = ');
  gotoxy(32,14);write('Valor de  X1 = ');
  gotoxy(32,16);write('Tolerancia t = ');
  gotoxy(21,24);write('Para continuar presione cualquier tecla');
  gotoxy(47,12);read(X0);
  gotoxy(47,14);read(X1);
  gotoxy(46,16);read(t);
  gotoxy(60,24);c := readkey;
end;

procedure Wengstein;
begin
  for i := 1 to 100 do
    begin
      X0 := (X1*G(X0) - X0*G(X1)) / ((X1-X0) - G(X1) + G(X0));
```

```
      if ABS(F(X0)) < T then
                        begin
                          mensaje := 1;
                          exit;
                        end;
      gotoxy(33,19);write('Procesando ...');
    end;
end;

procedure Resultado;
begin
  if mensaje = 1 then
     begin
       gotoxy(28,19);write('Raiz es:  x = ',X0:6:10);
       gotoxy(32,21);write('en ',i,' iteraciones.');
     end
    else
      begin
        gotoxy(25,19);write('No converge en: 100 iteraciones');
      end;
  gotoxy(21,24);write('Para continuar presione cualquier tecla');
  gotoxy(60,24);c:=readkey;
end;

begin
  Inicializa;
  Captura;
  Wengstein;
  Resultado;
end.
```

MÉTODO DE MÜLLER

Este método se obtiene mediante la expansión cuadrática de f(x) utilizando el polinomio fundamental de Newton.

Al aplicar la condición de raíz, x es raíz si f(x) = 0, se obtiene de este polinomio la fórmula correspondiente.

$$f(x) = f(x_0) + f[x_0,x_1](x-x_0) + f[x_0,x_1,x_2](x-x_0)(x-x_1)$$

igualando f(x) a cero :

$$0 = f(x_0) + f[x_0,x_1](x-x_0) + f[x_0,x_1,x_2](x-x_0)(x-x_1)$$

Supuestos los puntos x_0, x_1, x_2, esta última ecuación será una ecuación de segundo grado con respecto a x. La cual puede ser resuelta para evaluar dos posibles valores, de estos dos se seleccionará el que implique menor error de redondeo, considerando que se está aplicando el método para encontrar una raíz real.

Algoritmo:

Encontrar una solución a $f(x) = 0$ dadas tres aproximaciones x_0, x_1, x_2:

Entrada: x_0, x_1, x_2 aproximaciones iniciales y la tolerancia t.

Salida: x solución aproximada, no converge.

Paso 1 para i = 1, ..., n hacer | - 8

Paso 2

$$A = \frac{\dfrac{f(x_2) - f(x_1)}{x_2 - x_1} - \dfrac{f(x_1) - f(x_0)}{(x_1 - x_0)}}{(x_2 - x_0)}$$

$$B = \frac{f(x_2) - f(x_1)}{(x_2 - x_1)} + (x_2 - x_1)A$$

$$C = f(x_2) + x_2\left[\frac{f(x_1) - f(x_0)}{(x_1 - x_0)}\right] + x_2x_1A$$

Paso 3 $D = B^2 - 4ac$

Paso 4 Si $D<0$ entonces será aritmética compleja

Paso 5 $x = (-B + \sqrt{D})/2A$

Paso 6 Si $| f(x) | < t$ entonces Salida (x), Terminar.

Paso 7 Si $| f(x_1) | > | f(x_2) |$ entonces

Si $| f(x_1) | > | f(x_0) |$ entonces $x_1 = x$. Paso 1

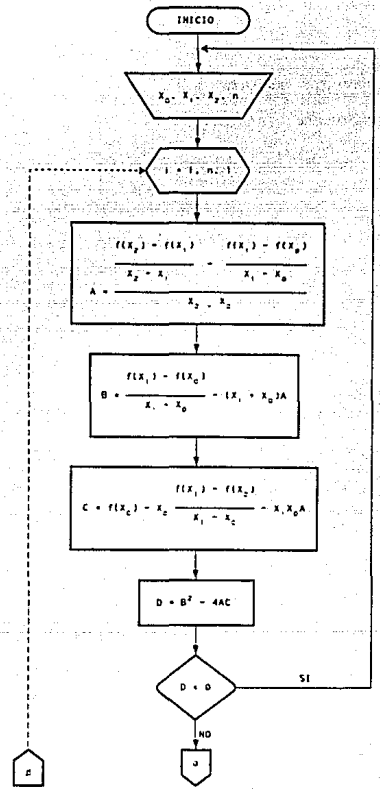
Paso 8 Si $| f(x_2) | > | f(x_0) |$ entonces $x_2 = x$. Paso 1

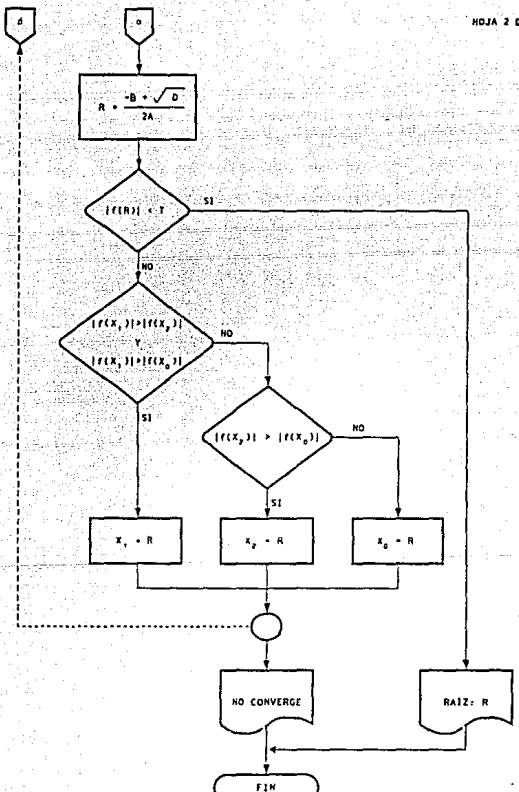
sino $x_0 = x$. Paso 1

Paso 9 Salida (No converge)

Terminar.

METODO DE MULLER




δ
α
R = (-B + √D) / 2A
|f(R)| < T
SI
NO
|f(X1)|>|f(X2)| Y |f(X1)|>|f(X0)|
NO
SI
|f(X2)| > |f(X0)|
NO
SI
X1 = R
X2 = R
X0 = R
NO CONVERGE
RAIZ: R
FIN

```
Program  Metodo_de_Muller;
uses crt;
var A1, A2, A, B, C, D, T, X, X0, X1, X2 : real;
    Solucion : byte;
    i, n : integer;
    cont : char;

function f(x : real) : real;
begin
   f := x*x + 4*x -2;
end;

procedure Inicializa;
begin
   A := 0;
   B := 0;
   C := 0;
   D := 0;
   i := 0;
   n := 0;
   x := 0;
   t := 0;
   A1 := 0;
   A2 := 0;
   X0 := 0;
   X1 := 0;
   X2 := 0;
   cont := char(0);
   Solucion := 0;
end;


procedure Captura;
begin
  clrscr;
  gotoxy(33,5);write('METODO DE MULLER');
  gotoxy(23,8);write('Funcion a evaluar F(x) = x^2 +4x -2');
  gotoxy(24,10);write('Suministre los siguientes datos :');
  gotoxy(30,13);write('Valor inicial (X0) = ');
  gotoxy(30,15);write('Valor inicial (X1) = ');
  gotoxy(30,17);write('Valor inicial (X2) = ');
  gotoxy(30,20);write('Tolerancia    (t)  = ');
  gotoxy(20,24);write('Presione cualquier tecla para continuar:');
  gotoxy(51,13);read(X0);
  gotoxy(51,15);read(X1);
  gotoxy(51,17);read(X2);
  gotoxy(51,20);read(T);
  gotoxy(60,24);cont := readkey;
end;
```

```
procedure Muller;
begin
   for i := 1 to 100 do
     begin
      writeln(i);
      A1 := (f(X1) - f(X0)) / (X1-X0);
      A2 := (f(X2) - f(X1)) / (X2-X1);
      A  := (A2 - A1) / (X2 - X0);
      B  := A1 - (X1 + X0)*A;
      C  := f(X0) - X0*A1 - X1*X0*A;
      D  := B*B - 4*A*C;
      if D < 0 then Captura;
      x  := (-B + sqrt(D)) / 2*A;
      if ABS(f(X)) < 1e-5 then exit;
      if (ABS(f(X1)) > ABS(F(X2))) or
        (ABS(f(X1)) > ABS(f(X0))) then X1 := x;
      if (ABS(f(X2)) > ABS(f(X0))) then X2 := x
                                   else X0 := x;
     end;
end;

procedure Resultado;
begin
  clrscr;
  gotoxy(33,5);write('METODO DE MULLER');
  gotoxy(23,8);write('Función evaluada F(x) = x^2 +4x -2');
  gotoxy(18,12);write('La raiz obtenida es  x = ',x);
  gotoxy(31,14);write('con ',i,' iteraciones');
  gotoxy(20,24);write('Presione cualquier tecla para continuar:');
  gotoxy(60,24);cont := readkey;
end;

begin
  Inicializa;
  Captura;
  Muller;
  Resultado;
end.
```

SUBSTITUCIÓN DIRECTA

El método de la substitución directa, punto fijo o iteración funcional como también se le conoce, se usa para obtener la solución de una ecuación $f(x) = 0$, comienza con un estimado e iterativamente mejora esta aproximación hasta que se obtiene precisión requerida.

Para algunas ecuaciones, puede ser difícil el encontrar un intervalo base, en estas circunstancias, es conveniente usar un método en el cual no se requiera ninguna información inicial. La iteración de punto fijo cumple este requerimiento.

Encontrar una raíz de una ecuación f es equivalente a encontrar un punto fijo de una ecuación g. Asumimos que la ecuación $f(x) = 0$ puede ser rearreglada como

$$x = g(x)$$

Así, a alguna solución de esta ecuación se le llama punto fijo de la ecuación g. Una iteración obvia para probar el cálculo de puntos fijos es

$$x_{n+1} = g(x_n), \qquad n = 0, 1, 2, \ldots$$

el valor de x_0 es elegido arbitrariamente y la posibilidad esta en que la secuencia $x_0, x_1, x_2, \ldots$, converja a un número α el cual automáticamente satisfaga a $x = g(x)$. Por otra parte, aunque $x = g(x)$ es un arreglo de $f(x) = 0$, α será una raíz de la ecuación.

En general, hay muchas diferentes maneras de rearreglar $f(x) = 0$ a la forma $x = g(x)$. Sin embargo, solamente alguna de estas será idónea para originar iteraciones que tengan éxito.

Algoritmo

Encontrar una solución a $x = g(x)$ dada la aproximación inicial x_0, con $g(x) = x - f(x)$:

Entrada x_0 aproximación inicial

Salida x solución aproximada

Paso 1 para i = 1, ..., n hacer 2, 3 y 4

Paso 2 $x = g(x)$

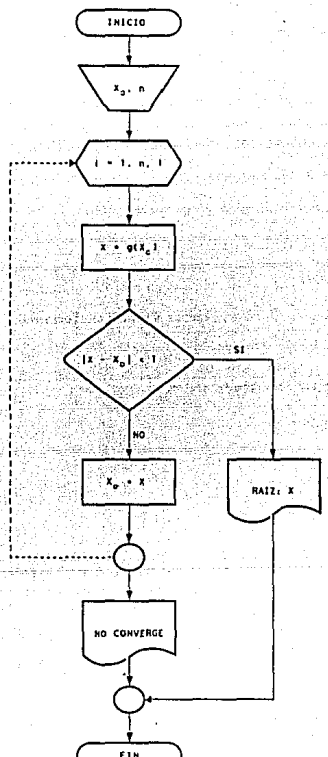
Paso 3 Si $|x - x_0| < t$ entonces Salida (x), Terminar.

Paso 4 $x_0 = x$

Paso 5 Salida (No converge)

Terminar.

METODO DE SUBSTITUCION DIRECTA



```
program Metodo_de_Sustitucion_Directa;
uses Crt;

var X0, x, t : real;
    i : Integer;
    c : char;
    mensaje : byte;

function F(x : real) : real;
begin
  F := exp(-x);
end;

procedure Inicializa;
begin
  X0 := 0;
  x  := 0;
  i  := 0; t := 0;
  mensaje := 0;
  c  := char(0);
end;

procedure Captura;
begin
  clrscr;
  gotoxy(31,5);write('SUSTITUCION DIRECTA');
  gotoxy(26,7);write('Función evaluada: Exp(-X) - X');
  gotoxy(25,10);write('Suministre los siguientes datos');
  gotoxy(29,14);write('Valor inicial:  x = ');
  gotoxy(32,16);write('Tolerancia: t = ');
  gotoxy(21,24);write('Para continuar presione cualquier tecla');
  gotoxy(49,14);read(X0);
  gotoxy(48,16);read(t);
  gotoxy(60,24);c:=readkey;
end;

procedure PuntoFijo;
begin
  for i := 1 to 100 do
    begin
      gotoxy(33,19);write('Procesando ...');
      x := f(X0);
      if ABS(x - X0) < T then
                        begin
                          mensaje := 1;
                          exit;
                        end;
      X0 := x;
    end;
end;
```

```
procedure Resultado;
begin
  if mensaje = 1 then
     begin
       gotoxy(28,19);write('Raiz es: x = ',x:6:10);
       gotoxy(32,21);write('en ',i,' iteraciones.');
     end
   else
     begin
       gotoxy(25,18);write('No converge en: 100 iteraciones');
     end;
  gotoxy(21,24);write('Para continuar presione cualquier tecla');
  gotoxy(60,24);c:=readkey;
end;

begin
  Inicializa;
  Captura;
  PuntoFijo;
  Resultado;
end.
```

MÉTODO DE LA BISECCIÓN

Este método también se denomina método de Bolzano, o búsqueda binaria. Y se presenta así: dada una función continua de una variable real x, la cual tiene un valor negativo en x = a y un valor positivo en x = b, entonces sabemos que existe un punto entre a y b donde la función toma un valor de cero. Si bisectamos el intervalo y revisamos en este punto si la función es positiva o negativa, entonces habremos encontrado un subintervalo en el que hay un cambio de signo (subintervalo en el cual existe una raíz). Al repetir el proceso de bisección, podremos acercarnos arbitrariamente a el cero.

Después de cada paso en el intervalo en donde se sitúa el cero es dividido a la mitad, diez iteraciones pueden reducir el intervalo en un factor de mil; veinte pasos en un millón; etc. Este método, el cual asume solamente continuidad y la capacidad para evaluar la función en cualquier punto, es bastante efectivo.

Algoritmo:

Encontrar una solución de $f(x) = 0$, dada la función continua f en el intervalo $[x_o, x_1]$ donde $f(x_o)$ y $f(x_1)$ tienen signos opuestos:

Entrada: x_o, x_1 extremos del intervalo.

Salida: x solución aproximada, mensaje no converge.

Paso 1 Si $f(x_o)*f(x_1) > 0$ entonces Entrada

Paso 2 para i = 1, ..., n realizar pasos 3, 4 y 5.

Paso 3 $x = (x_o + x_1) / 2$

Paso 4 Si $| f(x) | < t$ entonces Salida (x), Termina.

Paso 5 Si $f(x)*f(x_o) > 0$ entonces $x_o = x$, e ir al paso 2.

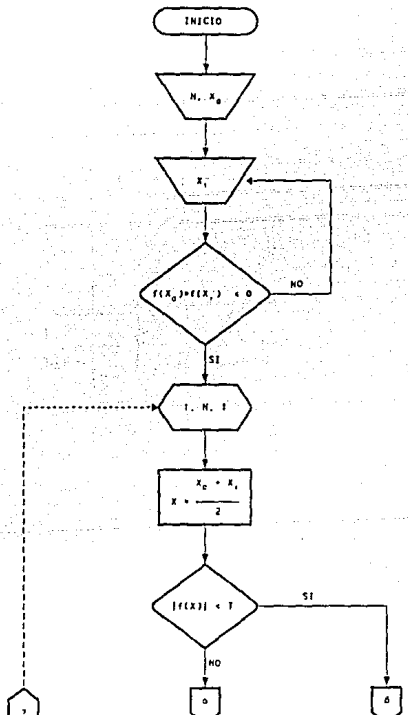
sino $x_1 = x$, e ir al paso 2.

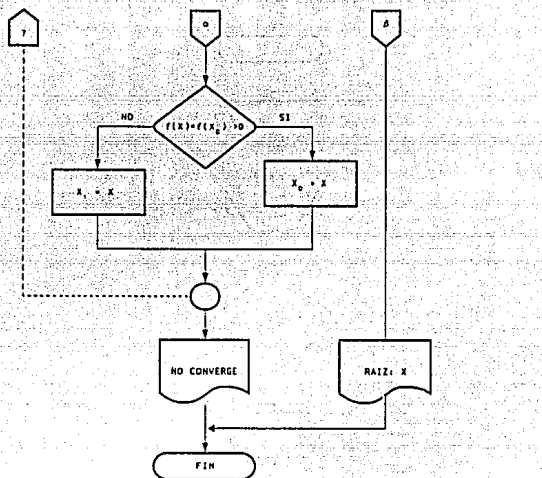
Paso 6 Salida (No converge)

Terminar.

siendo t la tolerancia, f(x), $f(x_o)$ la evaluación de la función en los puntos correspondientes.

METODO DE LA BISECCION




7
0
8
NO
f(X)*f(X0) >0
SI
X1 = X
X0 = X
NO CONVERGE
RAIZ: X
FIN

```
program Metodo_de_la_Biseccion;
uses Crt;

var X,X0,X1,t : real;
    i : Integer;
    c : Char;
    Solucion : boolean;

function F(X : real) : real;
begin
  F := X*X + 4*X -2;
end;

procedure Inicializa;
begin
 i := 0;
 t := 0;
 X := 0;
 X0:= 0;
 X1:= 0;
 Solucion := False;
end;

procedure Biseccion;
begin
  repeat
    i := succ(i);
    X := (X0 + X1)/2;
    if ABS(F(X)) < t then  Solucion := true;
    if F(X)*F(X0) > 0 then X0 := X
                      else X1 := X;
  until Solucion = true;
end;

procedure Captura;

procedure Cap_Desc;
begin
  gotoxy(51,16);read(X1);
  gotoxy(51,18);read(t);
  gotoxy(60,24);c := readkey;
  if F(X1)*F(X0) < 0 then Biseccion
                     else Cap_Desc;
end;

begin
  clrscr;
  gotoxy(30,5);writeln('METODO DE LA BISECCION');
  gotoxy(23,8);writeln('Función a evaluar F(x) = x^2 +4x -2');
  gotoxy(24,10);writeln('Suministre los siguientes datos :');
```

```
  gotoxy(30,14);write('Valor inicial (X0) = ');
  gotoxy(30,16);write('Valor inicial (X1) = ');
  gotoxy(30,18);write('Tolerancia    (t) = ');
  gotoxy(20,24);write('Presione cualquier tecla para continuar:');
  gotoxy(51,14);read(X0);
  Cap_Desc;
end;

procedure Resultado;
begin
  clrscr;
  gotoxy(30,5);write('METODO DE LA BISECCION');
  gotoxy(23,8);write('Función evaluada F(x) = x^2 +4x -2');
  gotoxy(18,12);write('La raíz obtenida es  x = ',X);
  gotoxy(31,14);write('con ',i,' iteraciones');
  gotoxy(20,24);write('Presione cualquier tecla para continuar:');
  gotoxy(60,24);c := readkey;
end;

begin
  Inicializa;
  Captura;
  Biseccion;
  Resultado;
end.
```

MÉTODO DE HOOKE-JEEVES

El presente método es otra técnica de búsqueda directa, el método parte de una aproximación o punto inicial x_0 de la función f(x), se incrementa este valor, a continuación se evalua en f(x) y se escoge aquel que hace menor el valor de la función de entre x_0, $x_0 + \Delta x$ y $x_0 + 2\Delta x$ para tomarlo como la siguiente estimación, esto es el método consta de exploraciones para encontrar el valor más idoneo y después tomarlo para realizar la siguiente estimación.

Algoritmo:

Encontrar una solución de $f(x) = 0$, dada la aproximación inicial x_0 y un incremento Δ_x:

Entrada: x_0 aproximación inicial, Δ_x incremento y la tolerancia t

Salida: x_0 solución aproximada, mensaje no converge.

Paso 1 Si $| f(x_0) | < t$ entonces Salida (x_0), **Terminar.**

Paso 2 $x_1 = x_0 + \Delta_x$

Paso 3 Si $| f(x_1) | > | f(x_0) |$ entonces

$x_1 = x_0 - 2\Delta_x$

Si $| f(x_1) | > | f(x_0) |$ entonces $\Delta_x = \Delta_x/2$

Si $\Delta_x > t$ entonces paso 2.

sino paso 8.

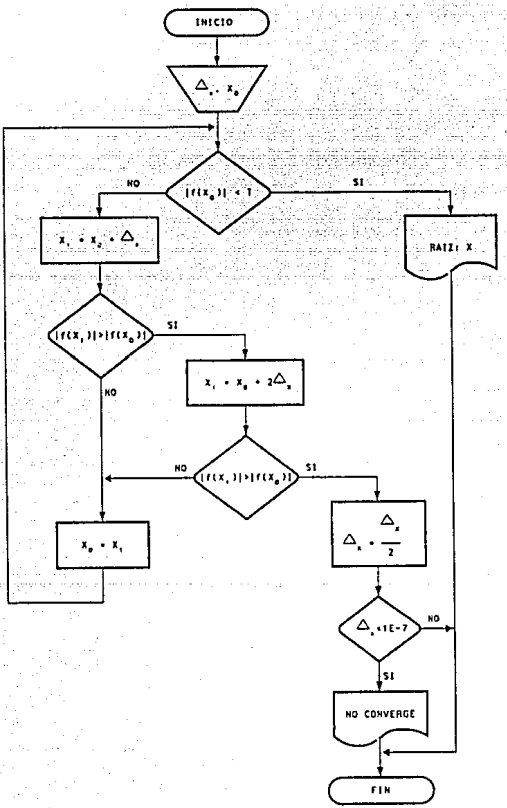
Paso 7 $x_0 = x_1$, e ir al paso 1.

Paso 8 **Salida** (No converge)

Terminar.

siendo t la tolerancia, $f(x_0)$, $f(x_1)$ la evaluación de la función en los puntos correspondientes y Δ_x el incremento.

METODO DE HOOKE-JEEVES



```
Program Metodo_de_Hooke_Jeeves;
uses Crt;

var X0, X1, inc, t : real;
    i : Integer;
    c : Char;
    Solucion : boolean;

function F(X : real) : real;
begin
  F := X*X + 4*X -2;
end;

procedure Inicializa;
begin
 i   := 0;
 t   := 0;
 X0  := 0;
 X1  := 0;
 inc := 0;
 Solucion :=  false;
 c := Char(0);
end;

procedure Captura;
begin
  clrscr;
  gotoxy(30,5);write('METODO DE HOOKE-JEEVES');
  gotoxy(23,8);write('Función a evaluar F(x) = x^2 +4x -2');
  gotoxy(24,10);write('Suministre los siguientes datos :');
  gotoxy(30,14);write('Valor inicial (X0) = ');
  gotoxy(30,16);write('Incremento    (dx) = ');
  gotoxy(30,18);write('Tolerancia    (t)  = ');
  gotoxy(20,24);write('Presione cualquier tecla para continuar:');
  gotoxy(51,14);read(X0);
  gotoxy(51,16);read(inc);
  gotoxy(51,18);read(t);
  gotoxy(60,24);c := readkey;
end;
procedure Hooke_Jeeves;
begin
  repeat
    i := i + 1;
    X1 := X0 + inc;
    if ABS(F(X1)) < ABS(F(X0)) then
         begin
           X0 := X1;
           if abs(F(X0)) < T then Solucion := true;
           gotoxy(33,21);write('Procesando ...');
         end
```

```
        else
          begin
            X1 := X0 - 2*INC;
            if ABS(F(X1)) < ABS(F(X0)) then
              begin
                X0 := X1;
                if abs(F(X0)) < T then Solucion := true;
                gotoxy(33,21);write('Procesando ...');
              end
            else
              begin
                INC := INC/2;
                if (Solucion = true) then
                  begin
                    if INC <= 1E-7 then
                      begin
                        clrscr;
                        gotoxy(15,20);
                        write('No converge el método, se ');
                        gotoxy(41,20);
                        write('realizaron ',i,' iteraciones');
                        gotoxy(60,24);
                        c := readkey;
                        Solucion := false;
                        exit;
                      end;
                  end
                else
                  Hooke_Jeeves;
              end;
          end;
      until Solucion <> false;
    end;
    procedure Resultado;
    begin
      clrscr;
      gotoxy(30,5);write('METODO DE HOOKE-JEEVES');
      gotoxy(23,8);write('Función evaluada F(x) = x^2 -4x -2');
      gotoxy(18,12);write('La raíz obtenida es  x = ',X0);
      gotoxy(31,14);write('con ',i,' iteraciones');
      gotoxy(20,24);write('Presione cualquier tecla para continuar:');
      gotoxy(60,24);c := readkey;
    end;

    begin
      Inicializa;
      Captura;
      Hooke_Jeeves;
      Resultado;
    end.
```

ECUACIÓN CUADRATICA

Las raíces de una ecuación cuadrática $ax^2 + bx + c = 0$ pueden ser evaluadas con la fórmula general de la ecuación cuadrática.

$$x = \frac{b^2 \pm \sqrt{b^2 - 4ac}}{2a}$$

la cual es probablemente la mejor manera de evaluar raíces complejas. Sin embargo, este no siempre es el mejor medio para determinar valores numéricos de las raíces cuando estas son reales, particularmente si la relación de las raíces es grande (b^2 - 4ac).

Un método más práctico en muchos casos es un proceso iterativo basado en el uso de las relaciones

$$x_1 + x_2 = -\frac{b}{a}, \qquad x_1 x_2 = \frac{c}{a} \tag{1}$$

donde x_1 y x_2 son las dos raíces, Si x_1 es la raíz de mayor módulo, entonces las aproximaciones sucesivas a la raíz pueden ser evaluadas utilizando la fórmula

$$x_1 = -\frac{b}{a} - x_2, \qquad x_2 = \frac{c}{a x_1} \tag{2}$$

alternativamente, iniciando con la aproximación $x_2 = 0$ si no hay otra disponible.

Este proceso, aún cuando es de primer orden, puede ser incluido fácilmente cuando $b^2 >> 4ac$, esta converge tan rápidamente, que es innecesario depurarla más. Si b^2 no es

considerablemente mayor a 4ac, puede ser conveniente usar el proceso de segundo orden derivado del proceso de primer orden.

La eliminación de x_1 entre las ecuaciones (2) da

$$x_1 = -\frac{b}{a} - \frac{(\frac{c}{a})}{x_1}$$

la cual es la expresión general.

Algoritmo

Encontrar una solución de $f(x) = ax^2 + bx + c = 0$, dados los coeficientes a, b, c de la ecuación cuadrática:

Entrada a, b, c coeficientes de la ecuación cuadratica.

Salida x_1, x_2 raíces reales. $x_1 \pm x_2$ raíces complejas.

Paso 1 $D = b^2 - 4ac$

Paso 2 Si $D => 0$ entonces

Tomar $x_1 = (-b + \sqrt{D})/2a$

Tomar $x2 = (-b - \sqrt{D})/2a$, paso 3

sino

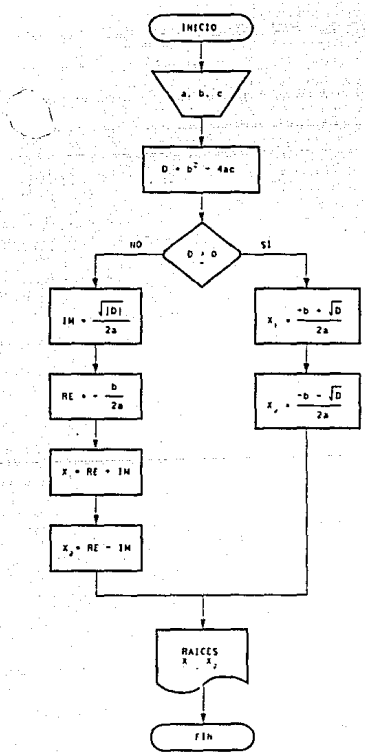
Tomar $x_1 = -b/2a$

Tomar $x_2 = (\sqrt{|d|})/2a$

Paso 3 Salida (x_1, x_2 o $x_1 \pm x_2$)

Terminar.

ECUACION CUADRATICA



```
Program Ecuacion_cuadratica;
uses Crt;
var A,B,C,D,X1,X2,XI,XR : real;
    co : Char;
begin
  clrscr;
  gotoxy(22,3);write('RESOLUCION DE ECUACIONES CUADRATICAS');
  gotoxy(20,6);WRITE('Suministre el valor de los coeficientes:');
  gotoxy(35,10);write('A = ');
  gotoxy(35,12);write('B = ');
  gotoxy(35,14);write('C = ');
  gotoxy(40,10);readln(A);
  gotoxy(40,12);readln(B);
  gotoxy(40,14);readln(C);
  D := B*B - 4*A*C;
  if D >= 0 then
       begin
         X1 := (-B + SQRT(D)) / (2*A);
         X2 := (-B - SQRT(D)) / (2*A);
         gotoxy(33,18);write('RAICES REALES');
         gotoxy(25,20);write('1ª raiz:  x = ',X1);
         gotoxy(25,22);write('2ª raiz:  x = ',X2);
       end
     else
       begin
         XI := SQRT(ABS(D))/2;
         XR := -B/(2*A);
         gotoxy(32,18);write('RAICES COMPLEJAS');
         if D > 0 then
           begin
            gotoxy(15,20);write('1ª raiz x = ',XR,' + ',XI,' i');
            gotoxy(15,22);write('2ª raiz x = ',XR,' - ',XI,' i');
           end
          else
           begin
            XI := ABS(XI);
            gotoxy(15,20);write('1ª raiz x = ',XR,' - ',XI,' i');
            gotoxy(15,22);write('2ª raiz x = ',XR,' + ',XI,' i');
           end;
       end;
  gotoxy(21,24);write('Presione cualquier tecla para continuar');
  gotoxy(60,24);co := readkey;
end.
```

ECUACIÓN CUBICA

Una ecuación cúbica con coeficientes reales tiene por lo menos una raíz real, es decir, x_1; si ésta está determinada, la división del cúbico por $(x - x_1)$ da una ecuación cuadrática la cual puede ser resuelta por la fórmula general de la ecuación cuadrática o por iteración. Para la determinación de la o las raíces reales, pueden utilizarse los métodos previos.

También existen métodos especiales disponibles, los cuales dependen de la reducción de la ecuación cúbica a una forma estándar.

Un método debido a Tartaglia-Cardano es el siguiente. Teniendo la ecuación cúbica:

$$ax^3 + 3bx^2 + 3cx + d = 0 \qquad (1)$$

tendrá tres raíces, la suma de las cuales valdrá (-3b/a); haciendo el cambio de variable:

$$z = x + \frac{b}{a} \qquad (2)$$

se obtiene la ecuación

$$z^3 + 3Hz + G = 0 \qquad (3)$$

en la que la suma de sus raíces es cero, y donde:

$$a^2H = ac - b^2$$

y

$$a^3H = a^2d - 3abc + 2b^3$$

para resolver la ecuación (3), se hace una substitución posterior, es decir:

$$z = u + v \qquad (4)$$

donde v es una función de u sin determinar todavía. Por tanto,

$$u^3 + v^3 + 3(uv + H)(u + v) + G = 0 \qquad (5)$$

substituyendo $v = -H/u$, la ecuación (5) se simplifica a

$$u^6 + Hu^3 + Gu^3 = 0 \qquad (6)$$

la ecuación (6) es de segundo grado, siendo la variable u cúbica, y tiene solución:

$$u^3 = -\frac{1}{2}G \pm \sqrt{(G^2 - 4H^3)} \qquad (7)$$

la ecuación determina seis valores para u, y puesto que $uv = H$ se determinan seis valores para v. Por consiguiente, la ecuación (4) parece dar seis raíces a la cúbica (3). Sin embargo, sólo hay tres valores distintos de z, debido a que cada par de valores de u y de v, dan dos valores iguales: u+v, v+u. La ecuación (2), transforma los valores de z en los correspondientes de x, que satisfacen a la ecuación (1).

En general, si la función cúbica (1) tiene tres raíces reales, $G^2 + 4H^3 < 0$, y la ecuación (6) tiene raíces complejas.

La raíz de la ecuación cúbica (7) se evalúa mejor en este caso en el diagrama de Argand. Cuando la ecuación cúbica tiene sólo una raíz real, la ecuación (6) tendrá siempre raíces reales. Por consiguiente, es el método más útil para encontrar una raíz real única de una ecuación cúbica.

Algoritmo:

Encontrar las raíces de $f(x) = x^3 + Ax^2 + Bx + C$, dados los coeficientes A, B, C, de la ecuación cúbica.

Entrada: A, B, C, coeficientes de la ecuación cúbica.

Salida: x_1, x_2, x_3 raíces reales, o x_1 real, x_2, x_3 complejas.

Paso 1 Hacer b1 = a1/3; b2 = a2/3; b3 = a3

Hacer

$$R = -\frac{b2 - 3b2b1 + 2b1^3}{2}$$

Hacer $Q = b2 - b1^2$

Hacer $D = R^2 + Q^3$

Paso 2 Si D<0 entonces paso 3

sino paso 4

Paso 3 Hacer

$$sign = \sqrt{D + R^2}$$

Hacer θ = TG(R,D)

Realizar desde k = 0 hasta 2

$$f_{k+1} = 2Rc(sign) \cdot \cos\left(\frac{\theta + 2k\pi}{3}\right) - b1$$

Hacer metodo = 1, e ir a paso 5

Paso 4 Hacer X1 = Rc(R + $\sqrt{}$D) + Rc(R - $\sqrt{}$D) -b1

Hacer a4 = 3b1 + X1

a5 = a4*X1 + 3b2

a6 = a5*X1 + b3

w = a4² - 4a5

Si w = 0 entonces paso 4.1

sino paso 4.2

Paso 4.1 Hacer X2 = -a4/2

X3 = -a4/2

metodo = 2, e ir a paso 5

Paso 4.2 Si w > 0 entonces paso 4.21

sino paso 4.22

Paso 4.21 Hacer X2 = -a4/2 + $\sqrt{}$w

X3 = -a4/2 - $\sqrt{}$w

metodo = 3, e ir a paso 5

Paso 4.22 Hacer X2 = -a4/2

X3 = $\sqrt{}$ | w | /2

metodo = 4, e ir a paso 5

Paso 5 En caso de que metodo valga

1: Para i = 1 hasta 3

Imprimir x_i, e ir a paso 6

2,3: Imprimir X1, X2, X3, e ir a paso 6

4: Imprimir X1, X2 ± X3, e ir paso 6

Paso 6 Terminar.

FUNCIONES

TG Calcula el valor de la tangente (La razón de esta función estriba en el error que tiene la función que provee Turbo Pascal al evaluar un ángulo que esta en el tercer y cuarto cuadrante).

Paso 1 Si R = 0 entonces paso 2

sino paso 3

Paso 2 Si $\sqrt{}$d<0 entonces TG = π/2, e ir a paso

sino Imprime error angulo no definido, paso 4

Paso 3 Si $\sqrt{}$d=0 entonces paso 3.1

sino paso 3.2

Paso 3.1 Si R>0 entonces TG = 0, e ir a paso 4

sino TG = π, e ir a paso 4

Paso 3.2 Hacer M = ARCTAN($\sqrt{}$ | D | /R

Si m=0 entonces Si $\sqrt{D}<0$ entonces m = m + π, paso 3.2

sino Si $\sqrt{D}>0$ entonces m = m + π, paso 3.3

sino m = 2mπ, paso 3.3

Paso 3.3 Hacer TG = m, e ir a paso 4

Paso 4 Regresa.

Calcula la raíz del argumento SIGN

Paso 1 Si SIGN = 0 entonces Rc = 0, e ir a paso 3

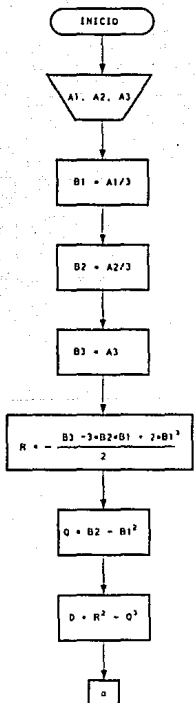
Paso 2 Hacer s = 1

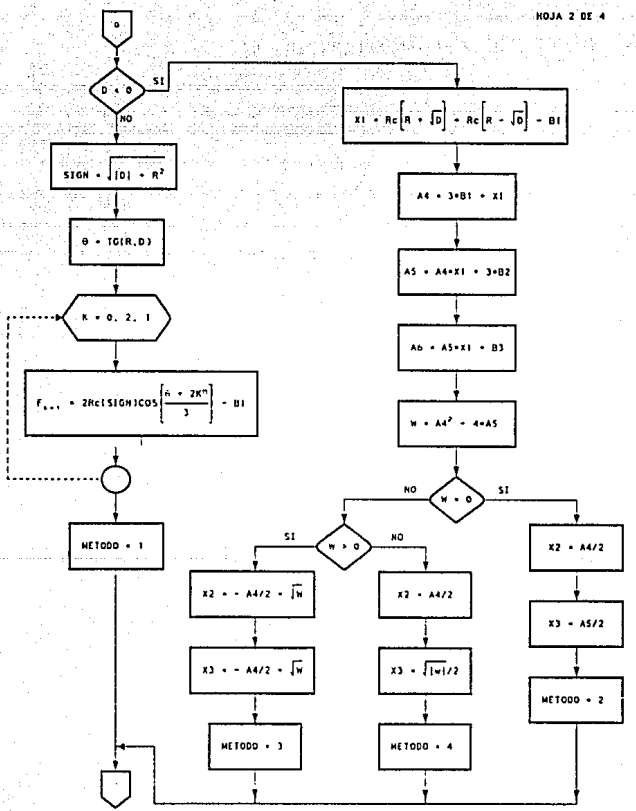
Si SIGN < 0 entonces s = -1

Hacer Rc = s*Exp(Ln | SIGN |)/3, e ir a paso 3

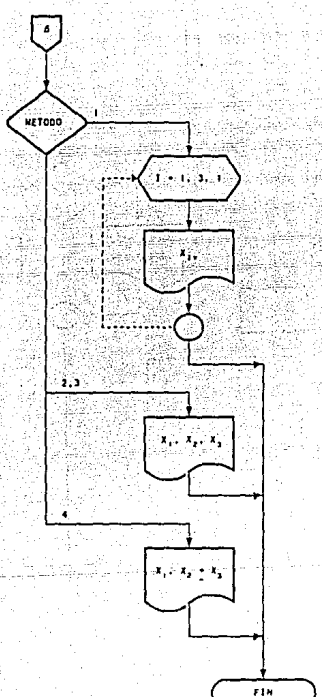
Paso 3 Regresa.

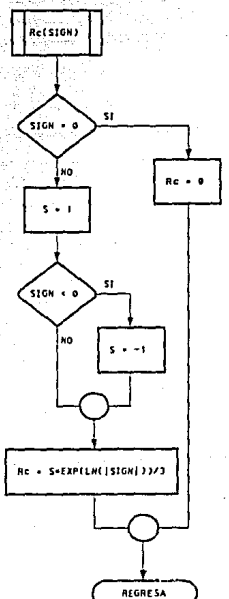
METODO DE CARDANO-TARTAGLIA



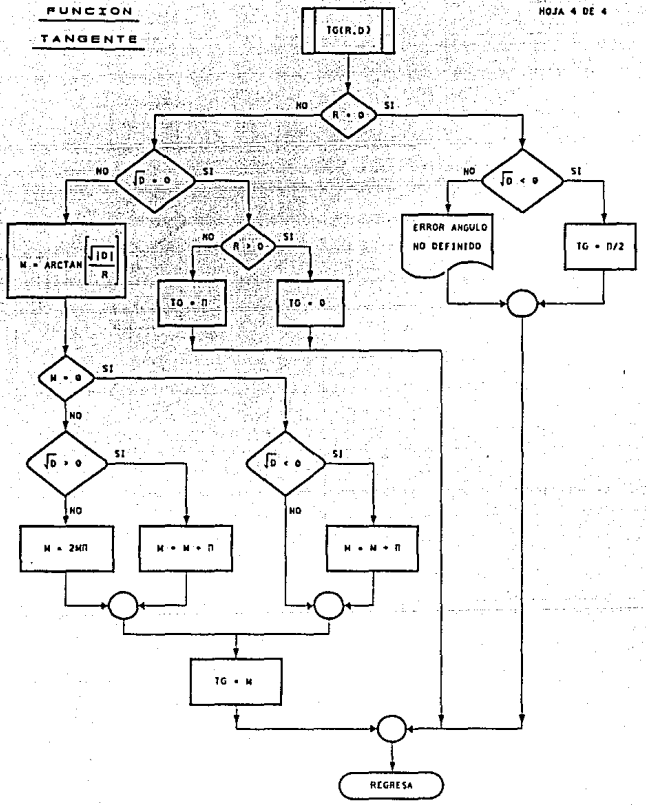

a
D < 0
SI
NO
X1 = Rc[R + √D] + Rc[R - √D] - B1
SIGN = √|D| + R²
A4 = 3*B1 + X1
θ = TG(R,D)
A5 = A4*X1 + 3*B2
K = 0, 2, 1
A6 = A5*X1 + B3
F... = 2Rc(SIGN)COS[(θ + 2Kπ)/3] - B1
W = A4² - 4*A5
W = 0
NO
SI
W > 0
SI
NO
X2 = A4/2
X3 = A5/2
METODO = 2
X2 = - A4/2 + √W
X3 = - A4/2 + √W
X2 = A4/2
X3 = √|W|/2
METODO = 1
METODO = 3
METODO = 4

RAIZ


6
METODO
1
I = 1, 3, 1
X1,
2,3
X1, X2, X3
4
X1, X2 = X3
FIN


Rc(SIGN)
SIGN = 0
SI
NO
Rc = 0
S = 1
SIGN < 0
SI
NO
S = -1
Rc = S*EXP(ALOG(|SIGN|))/3
REGRESA

FUNCION TANGENTE



```
Program Metodo_de_Cardano_Tartaglia;
uses crt;
var  i, k, metodo : byte;
     c : char;
     a1, a2, a3, a4, a5, a6 : real;
     b1, b2, b3, D, d1, d2  : real;
     X1, X2, X3, p, Q, R, w : real;
     x, teta, sign : real;
     f : array[1..3] of real;
procedure Inicializa;
begin
  k := 0;
  a1 := 0;
  a2 := 0;
  a3 := 0;
  a4 := 0;
  a5 := 0;
  a6 := 0;
  b1 := 0;
  b2 := 0;
  b3 := 0;
  D  := 0;
  d1 := 0;
  d2 := 0;
  X1 := 0;
  X2 := 0;
  X3 := 0;
  p  := 0;
  Q  := 0;
  R  := 0;
  w  := 0;
  x  := 0;
  sign  := 0;
  teta := 0;
  metodo := 0;
  for i := 1 to 3 do
     f[i] := 0;
end;

function Rc(sign : real) : real;
var s : integer;
begin
   if sign = 0 then Rc := 0
           else
             begin
               s:=1;
               if sign < 0 then s := -1;
               Rc := s*exp(ln(abs(sign))/3);
             end;
end;
```

```
function Tg(R, D : real) : real;
var m : real;
begin
  if R = 0 then
     begin
       if sqrt(abs(D))>0 then Tg := pi/2
         else
           if sqrt(abs(D)) < 0 then Tg:=3*pi/2
            else
              begin
                gotoxy(30,15);write('Error: θ no definida');
                gotoxy(17,17);write('Presiona cualquier tecla');
                gotoxy(41,17);write(' para salir del programa');
                c := readkey;
                halt;
              end;
            exit;
     end;
  if sqrt(abs(D))=0 then
     begin
       if R > 0 then tg:=0
         else tg := pi;
       exit;
     end;
  m := arctan(sqrt(abs(D))/R);
  if m = 0 then
          if sqrt(abs(D))<0 then m:=m+pi
        else begin
               if sqrt(abs(D))>0 then m:=m+pi
                                 else m:=2*pi+m;
               tg:=m
             end;
end;
procedure Cardano_Tartaglia;
begin
  b1:=a1/3;
  b2:=a2/3;
  b3:=a3;
  R:=-0.5*(b3-3*b2*b1+2*b1*b1*b1);
  Q:=b2-b1*b1;
  D:=R*R+Q*Q*Q;
  if D < 0 then
           begin
             sign:=sqrt(abs(D)+R*R);
             teta := Tg(R,D);
             for k:=0 to 2 do
               begin
                 t[k+1] := 2*Rc(sign)*cos((teta+k*2*pi)/3)-b1;
                 metodo := 1;
               end;
```

```
            end
          else
            begin
              X1 := Rc(R+sqrt(D))
                  + Rc(R-sqrt(D))-b1;
              a4:=3*b1+X1;
              a5:=a4*X1+3*b2;
              a6:=a5*X1+b3;
              begin
                w:=(a4*a4-4*a5);
                if w = 0 then
                    begin
                      X2 := -a4/2;
                      X3 := -a4/2;
                      metodo := 2;
                    end
                  else
                    begin
                     if w > 0 then
                          begin
                            X2 := -a4/2+sqrt(w);
                            X3 := -a4/2-sqrt(w);
                            metodo := 3;
                          end
                        else
                          begin
                            X2:=-a4/2;
                            X3:=sqrt(abs(w))/2;
                            metodo := 4;
                          end
                    end
              end
            end;
end;

procedure Captura;
begin
  textcolor(10);textbackground(5);
  clrscr;
  gotoxy(27,2);writeln('Metodo de Cardano Tartaglia');
  gotoxy(15,5);
  writeln('Solución de la ecuación cúbica: x^3 + Ax^2 + Bx + C');
  gotoxy(20,7);
  writeln('Suministre los coeficientes de la ecuación');
  gotoxy(30,10);writeln('A = ');
  gotoxy(40,10);writeln('B = ');
  gotoxy(50,10);writeln('C = ');
  gotoxy(21,24);writeln('Presione cualquier tecla para continuar');
  gotoxy(34,10);readln(a1);
  gotoxy(44,10);readln(a2);
```

```
  gotoxy(54,10);read(a3);
end;

procedure Resultado;
begin
  gotoxy(22,13);writeln ('Las raices de la ecuacion cubica son:');
  case metodo of
  1 : begin
        for i := 1 to 3 do
          begin
            gotoxy(33,15+i);writeln('X',i,' = ',r[i]);
          end;
      end;
  2,3:begin
        gotoxy(33,16);writeln('X[1] = ',X1);
        gotoxy(33,17);writeln('X[2] = ',X2);
        gotoxy(33,18);writeln('X[3] = ',X3);
      end;
  4 : begin
        gotoxy(33,16);write('X[1] = ',X1);
        gotoxy(23,17);write('X[2] = ',X2,' + ',abs(X3),'i');
        gotoxy(23,18);write('X[3] = ',X2,'- ',abs(X3),'i');
      end;
  end;
  gotoxy(60,24);c := readkey;
end;

begin
  Inicializa;
  Captura;
  Cardano_Tartaglia;
  Resultado;
end.
```

DIVISIÓN SINTÉTICA

Cuando f(x) es un polinomio de grado n, tal que la solución es una ecuación algebraica, los métodos anteriores pueden sistematizarse utilizando la división sintética.

Frecuentemente se tiene que dividir un polinomio entre un binomio de la forma x - a. Revisando cuidadosamente el procedimiento para realizar cada una de esas divisiones, puede verse que es posible hacer simplificaciones hasta llegar a la división sintética.

Sea

$$Q(x) = A_0x^{n-1} + A_1x^{n-2} + A_2x^{n-3} + \ldots + A_{n-2}x + A_{n-1}$$

el cociente y R el residuo que resulta de dividir el polinomio

$$P(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \ldots + a_{n-1}x + a_n$$

entre el binomio (x-a)Q(x) + R. O sea

$$a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \ldots + a_{n-1}x + a_n = A_0x^n + (A_1 - aA_0)x^{n-1} + (A_2 - aA_1)x^{n-2} + \ldots + (A_{n-1} - aA_{n-2})x + (R + aA_{n-1})$$

como los polinomios de ambos miembros son idénticos, los coeficientes de las mismas potencias

de x en ambos polinomios deben ser iguales entre si, luego:

$$
\begin{aligned}
a_0 &= A_0 \\
a_1 &= A_1 \\
a_2 &= A_2 \\
&\cdots\cdots\cdots\cdots \\
a_{n-1} &= A_{n-1} + aA_{n-2} \\
a_n &= R - aA_{n-1}
\end{aligned}
$$

de donde se obtiene

$$
\begin{aligned}
A_0 &= a_0 \\
A_1 &= a_1 + aA_0 \\
A_2 &= a_2 + aA_1 \\
&\cdots\cdots\cdots\cdots \\
A_{n-1} &= a_{n-1} + aA_{n-2} \\
R &= a_n + aA_{n-1}
\end{aligned}
$$

que son los coeficientes del polinomio cociente y el residuo buscados. Para determinar éstos usando las fórmulas anteriores, se pueden rearreglar los cálculos en la siguiente forma

a	a_0	a_1	a_2	$\cdots$	a_{n-1}	a
		aA_0	aA_1	$\cdots$	aA_{n-2}	aA_{n-1}
	A_0	A_1	A_2	$\cdots$	A_{n-1}	R

1. En el renglón, colocar todos los coeficientes de P(x) en forma descendente por las potencias de x, incluyendo los que valen cero.
2. Fuera, en la primera línea a la derecha de los coeficientes, escribir el valor de a, el divisor > > sintético > > si la división es entre x-a.
3. Bájese al tercer renglón el coeficiente dominante a_0 que es igual a A_0.
4. Multipliquese a por A_0: escribir el producto aA_0 en la segunda línea, abajo de a_1, súmense con a_1 y escríbase la suma de $aA_0 + a_1$ en la tercera línea bajo de a_1.

5. Multiplíquese la suma del paso 4 por a_1, escríbase el producto en la segunda línea bajo de a_2 súmese con a_2, y escríbase la suma en la tercera línea debajo de a_2.

6. Repítase el proceso del paso 5 hasta que haya sumado un producto al término constante a_n.

Los primeros n números de la tercera línea son los coeficientes del cociente, que es un polinomio de grado n-1, y el último número de la tercera línea es el residuo de f(a).

Algoritmo:

Reducción del grado de un polinomio p(x) dada una aproximación a una de sus raices Z y los coeficientes del polinomio:

Entrada: n grado del polinomio, Z aproximación a la raíz

a, coeficientes del polinomio

Salida: b_n residuo

Paso 1 $b_0 = a_n$

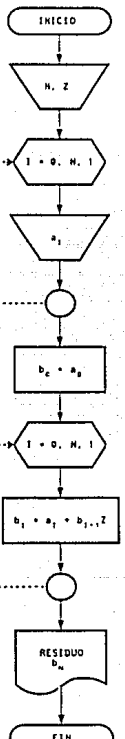
Paso 2 para i = 1, ..., n; hacer 3

Paso 3 $b_i = a_i + b_{i-1}Z$

Paso 4 Salida (b_n)

Terminar.

METODO DE LA DIVISION SINTETICA



```
Program Division_sintetica;
uses Crt;
var z : real;
    N,i : integer;
    A,B : array [1..10] of real;
    c : char;

procedure Inicializa;
begin
  z := 0;
  N := 0;
  I := 0;
  c := char(0);
end;

procedure Captura;
begin
  clrscr;
  gotoxy(30,3);write('DIVISION SINTETICA');
  gotoxy(25,5);write('Suministre los siguientes datos');
  gotoxy(21,8);write('Aproximacion (divisor sintético) Xo = ');
  gotoxy(22,10);write('Número de términos del polinomio N = ');
  gotoxy(59,8);read(z);
  gotoxy(59,10);read(n);
  for i := 1 to N do
    begin
      gotoxy(31,10+i*2);write('Coeficiente A',i-1,' = ');
      gotoxy(49,10+i*2);read(A[i]);
    end;
  gotoxy(21,24);write('Presione cualquier tecla para continuar');
  gotoxy(60,24);c := readkey;
end;

procedure Divsintetica;
begin
  B[1] := A[1];
  for i := 2 to n do
    begin
      B[i] := A[i] + B[i-1]*z;
    end;
  gotoxy(27,22);write('Residuo = ',B[n]);
  gotoxy(21,24);write('Presione cualquier tecla para continuar');
  gotoxy(60,24);c := readkey;
end;

begin
  Inicializa;
  Captura;
  Divsintetica;
end.
```

CAPITULO IV

INTEGRACION

INTEGRACIÓN NUMÉRICA

Algunas veces es necesario realizar un cálculo que implica una integración. Por ejemplo, el gasto volumétrico de un gas a través de un ducto puede determinarse a partir de la distribución de velocidades lineales mediante la evaluación de la integral correspondiente. El valor medio de una cantidad particular, se obtiene frecuentemente mediante integración. Por ejemplo, puede determinarse una temperatura superficial media integrando la distribución superficial de temperaturas y dividiendo entre el área.

Una manera de integrar un conjunto de datos es encontrar una ecuación empírica para los puntos, utilizando uno de los métodos de ajuste de datos, y entonces integrar la ecuación analíticamente.

NOTA: La función a integrar se incluye como otra línea de programa en la sección de declaraciones, para integrar cualquier otra función diferente bastará con substituirla en dicha línea.

REGLA DEL TRAPECIO

La regla del trapecio es una de las fórmulas cerradas de Newton-Cotes. Consideremos la función f(x), cuya gráfica se muestra en la siguiente página.

Se puede obtener una aproximación al área bajo el segmento de curva entre los dos puntos base $x_0 = a$ y $x_1 = b$ subdividiendo el intervalo de a a b en n fajas de ancho

$$h = \frac{b - a}{n}$$

y aproximando el área de cada subintervalo mediante un trapecio, el cual se forma reemplazando la curva por su línea secante, trazada entre los puntos base. Entonces para evaluar

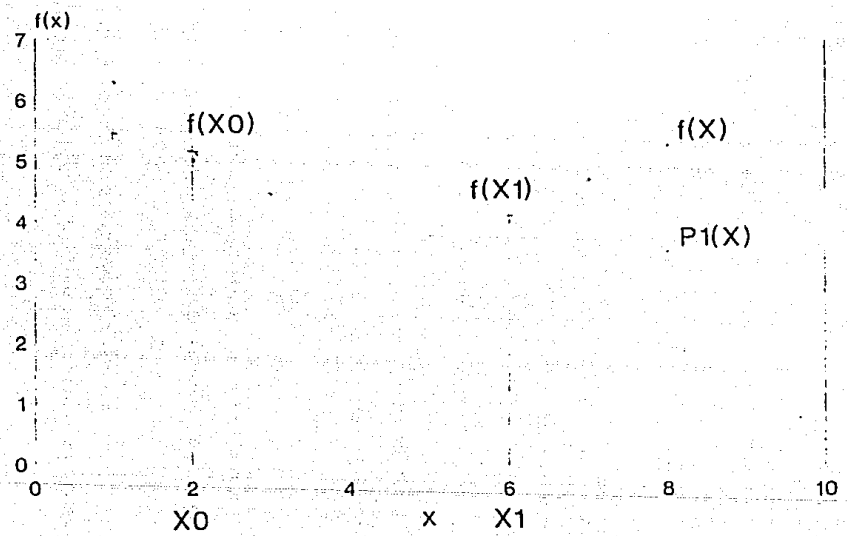
$$\int_a^b f(x)\,dx$$

nos basamos en la aproximación de f(x) en $x_0 = a$ y $x_1 = b$ por una línea recta, esto corresponde a un polinomio de primer grado. Ahora, una línea recta puede ser representada como

$$f(x) = f(a) + \frac{f(b) - f(a)}{b - a}(x - a)$$

considerando triángulos semejantes. El área bajo la línea recta es una aproximación de la integral de f(x) entre los límites a y b.

REGLA DEL TRAPECIO



$$\int_a^b f(x) = \int_a^b [f(a) + \frac{f(b) - f(a)}{b - a}(x - a)]\,dx$$

el resultado de la integración es

$$\int_a^b f(x) = \frac{f(a) - f(b)}{2}(b - a)$$

a esta ecuación se conoce como regla del trapecio.

Algoritmo

Aproximar la integral $I = \int_a^b f(x)dx$ dados el límite inferior a y el límite superior b de la integral y n número de subintervalos en que se divide el rango completo:

Entrada: x_o, x_n límites inferior y superior, n número de subintervalos.

Salida: Area aproximación a la integral.

Paso 1 $h = (x_n - x_o)/(n-1)$

Paso 2 area = $(f(x_n) - f(x_o))/2$

Paso 3 para i = 1, ..., n-1 hacer

Hacer $x_o = x_o + h$

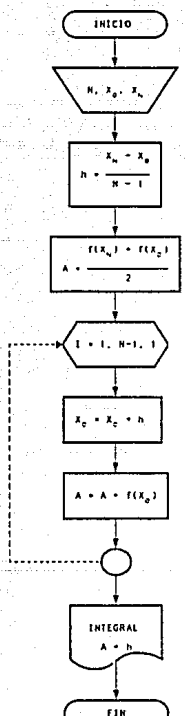
area = area + $f(x_o)$

Paso 4 area = area * h

Paso 5 Salida (area)

Terminar.

REGLA DEL TRAPECIO



```
Program Regla_del_Trapecio;
uses crt;

var a, h, x0, xn : real;
    i, n : byte;
    c : char;

function f(x : real) : real;
begin
    f := sqrt(x) + 4*x -2;
end;

procedure Inicializa;
begin
    a := 0;
    h := 0;
    i := 0;
    n := 0;
    c := char(0);
    x0 := 0;
    xn := 0;
end;

procedure Captura;
begin
    clrscr;
    Gotoxy(31,2);Write('REGLA DEL TRAPECIO');
    gotoxy(25,4);write('Función a integrar x^2 + 4x - 2');
    Gotoxy(25,5);Write('Suministre los siguientes datos');
    Gotoxy(30,8);Write('Límite inferior a = ');
    Gotoxy(30,10);Write('Límite superior b = ');
    Gotoxy(27,13);Write('Número de Intervalos n = ');
    Gotoxy(21,24);Write('Presione cualquier tecla para continuar');
    Gotoxy(50,8);Readln(x0);
    Gotoxy(50,10);Readln(xn);
    Gotoxy(52,13);Readln(n);
    Gotoxy(60,24);c := readkey;
end;

procedure Trapecio;
begin
    h := (xn - x0) / n;
    a := (f(xn) + f(x0))/2;
    for i := 1 to n-1 do
      begin
        x0 := x0 + h;
        a := a + f(x0);
      end;
    a := a*h;
end;
```

```
procedure Resultado;
begin
   Gotoxy(35,16);Writeln('RESULTADO');
   Gotoxy(30,19);Writeln('Integral:',a:11:7);
   Gotoxy(21,24);Write('Presione cualquier tecla para continuar');
   Gotoxy(60,24);C := readkey;
end;

begin
   Inicializa;
   Captura;
   Trapecio;
   Resultado;
end.
```

REGLA DE SIMPSON DE 1/3

Consideraremos ahora una de las técnicas de integración numérica más ampliamente conocidas y usadas, la regla de Simpson, de manera similar a la regla del trapecio, divide el intervalo total en intervalos menores y se aproxima el área de cada intervalo, la diferecia entre estos dos métodos es que hace pasar una parábola por las tres ordenadas de dos intervalos adyacentes. Se esperaría que, dado que la regla del trapecio es exacta para polinomios de primer grado, la regla de Simpson fuera exacta para polinomios de segundo grado o menor; en la realidad se obtiene el sorprendente resultado de que la regla de Simpson es exacta para polinomios de grado tres o menor. Por lo tanto, su alta precisión, en comparación con el esfuerzo que requiere y dado que su fórmula no es más complicada que la de la regla del trapecio, explican la amplia difusión del método.

Como ya se mencionó, el método consiste en conectar grupos sucesivos de tres puntos sobre la curva mediante parábolas de segundo grado, después sumar las áreas bajo las parábolas para obtener el área aproximada bajo la curva.

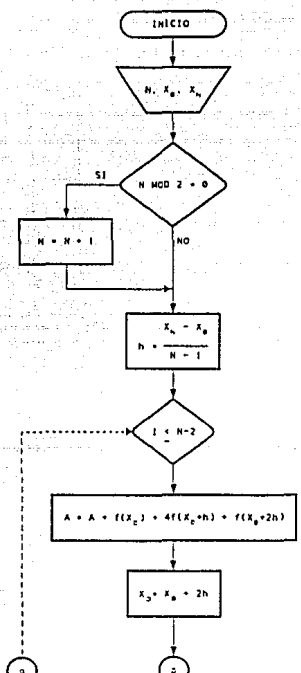
El uso de este método requiere que el número de subintervalos sea par, ya que la integración se hace tomando dos subintervalos cada vez.

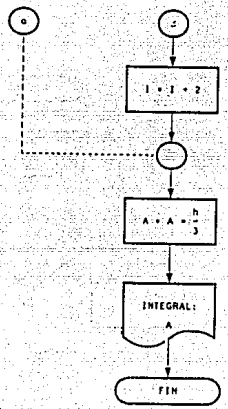
Algoritmo

Aproximar la integral $I = \int_{x_a}^{x_n} f(x)dx$ dados tanto el límite inferior como el superior de la integral y n número de subintervalos en que se divide el intervalo:

Entrada	x_a, x_n límites inferior y superior, n número de subintervalos.
Salida	Area aproximación a la integral.
Paso 1	Si el modulo 2 de n = 0 entonces n = n + 1
Paso 2	$h = (x_n - x_a)/(n-1)$
Paso 3	mientras i < = n-2 realiza
	area = area + $f(x_a) + 4f(x_a+h) + f(x_a+2h)$
	$x_a = x_a + 2h$
	i = i + 2
Paso 4	area = area * h/3
Paso 5	Salida (area)
	Terminar.

REGLA DE SIMPSON DE 1/3




0
2
I = I + 2
A = A + h/3
INTEGRAL:
A
FIN

```
Program Regla_de_Simpson_de_un_tercio;
uses  crt;
var a, b, Ar, h, x : real;
  i, j, n, m : integer;
  c : char;
procedure Inicializa;
begin
 a  := 0;
 b  := 0;
 Ar := 0;
 h  := 0;
 x  := 0;
 i  := 0;
 j  := 0;
 n  := 0;
 m  := 0;
 c  := char(0);
end;

function F(X : real) : real;
begin
 F := x*x+4*x-2;
end;

procedure Captura;
begin
 clrscr;
 gotoxy(28,3);write('REGLA DE SIMPSON DE 1/3');
 gotoxy(25,6);write('Función a integrar x^2 + 4x - 2');
 gotoxy(25,9);write('Suministre los siguientes datos');
 gotoxy(29,12);write('Valor inicial:   a = ');
 gotoxy(29,14);write('Valor final:     b = ');
 gotoxy(50,12);read(a);
 gotoxy(50,14);read(b);
 gotoxy(12,16);write('Número de puntos en el intervalo [',a:3:5,',',b:3:5,'] : n = ');
 gotoxy(21,24);write('Presione cualquier tecla para continuar');
 gotoxy(69,16);read(n);
 gotoxy(60,24);c := readkey;
end;

procedure Simpson13;
begin
 if n mod 2 = 0 then n := n + 1;
 h := (b - a)/(n-1);
 while j <= n-2 do
   begin
     Ar := Ar + F(a) + 4*F(a+h) + F(a+2*h);
     a := a + 2*h;
     j := j + 2;
   end;

 Ar := Ar * (h/3);
end;
```

```
procedure Resultado;
begin
 gotoxy(30,20);write('Area = ',At:10:6);
 gotoxy(21,24);writeln('Presione cualquier tecla para continuar');
 gotoxy(60,24);c := readkey;
end;

begin
 Inicializa;
 Captura;
 Simpson13;
 Resultado;
end.
```

REGLA SIMPSON DE 3/8

Otro método de integración es la regla de Simpson de 3/8. Este método consiste en obtener el área bajo la curva contenida en tres subintervalos consecutivos y sumando después éstas áreas para determinar el área total. El mecanismo utilizado en la regla de los tres octavos es similar al de un tercio, excepto porque se determina el área bajo una curva de tercer grado que conecta cuatro puntos sobre una curva dada. Este método se utiliza cuando el número de puntos tabulados excede en uno a un multiplo de tres mientras que la regla de un tercio requiere que el intervalo sea dividido en un número par de subintervalos. Frecuentemente se combinan ambos métodos, aplicando la regla de un tercio sobre el número de subintervalos menos tres, y completando la integración con la regla de tres octavos sobre los tres subintervalos restantes. Un método alternativo para un número impar de subintervalos, consiste en integrar uno de ellos mediante la regla del trapecio aunque la precisión obtenida es algo menor.

Algoritmo:

Aproximar la integral $I = \int_{x_0}^{x_n} f(x)dx$ dados tanto el límite inferior como el superior de la integral y n número de subintervalos en que se divide el intervalo:

Entrada: x_0, x_n límites inferior y superior, n número de subintervalos.

Salida: Area aproximación a la integral.

Paso 1 Si el módulo 3 de n-1 no es 0 entonces Lee nuevamente n y repite Paso 1

Paso 2 $h = (x_n - x_0)/(n-1)$

Paso 3 mientras $i < n-3$ realiza

$area = area + f(x_0) + 3f(x_0+h) + 3f(x_0+2h) + f(x_0+3h)$

$x_0 = x_0 + 3h$

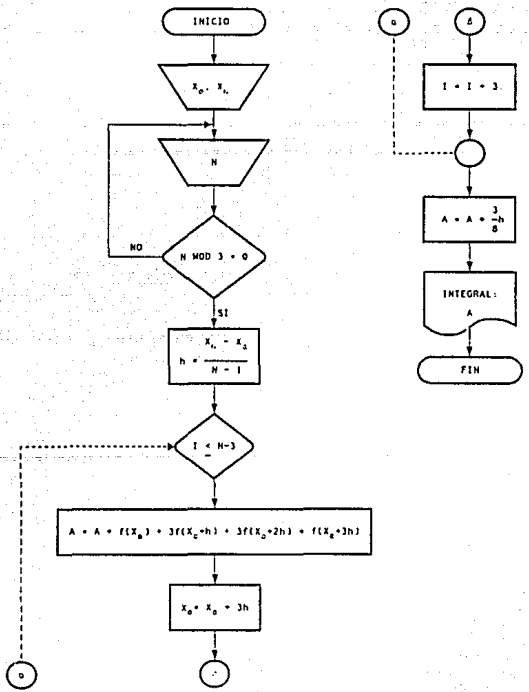
$i = i + 3$

Paso 4 $area = area * 3h/8$

Paso 5 Salida (area)

Terminar.

REGLA DE SIMPSON DE 3/8



```
Program Regla_de_Simpson_de_tres_octavos;
uses crt;
var a, b, Ar, h, x : real;
  i, k, n, m : integer;
  c : char;

procedure Inicializa;
begin
 a := 0;
 b := 2;
 Ar := 0;
 h := 0;
 x := 0;
 i := 0;
 k := 0;
 n := 0;
 m := 0;
 c := char(0);
end;

function F(X : real) : real;
begin
 F := x*x + 4*x - 2;
end;

procedure Captura;

procedure Inter;
begin
gotoxy(69,16);readln(n);
n := n + 1;
k := n - 1;
if k mod 3 <> 0 then
 begin
   gotoxy(69,16);clreol;
   gotoxy(18,18);Textcolor(0);textbackground(7);
   gotoxy(18,18);write('El número de intervalos debe ser multiplo de 3.');
   gotoxy(85,18);Textcolor(7);textbackground(0);
   gotoxy(60,24);c := readkey;
   gotoxy(1,18);Clreol;
   Inter;
 end;
end;

begin
 clrscr;
 gotoxy(28,3);write('REGLA DE SIMPSON DE 3/8');
 gotoxy(25,6);write('Función a integrar x^2 + 4x - 2');
 gotoxy(25,9);write('Suministre los siguientes datos');
 gotoxy(29,12);write('Valor inicial:  a = ');
```

```
 gotoxy(29,14);write('Valor final:     b = ');
 gotoxy(12,16);
 write('Número de puntos en el intervalo [',a:3:5,',',b:3:5,'] : n = ');
 gotoxy(21,24);write('Presione cualquier tecla para continuar');
 gotoxy(50,12);read(a);
 gotoxy(50,14);read(b);
 Inter;
 gotoxy(60,24);c := readkey;
end;

procedure Simpson38;
begin
 h := (b - a)/(n - 1);
 while i < n-3 do
  begin
    Ar := Ar + F(a) + 3*F(a+h) + 3*F(a+2*h) + F(a+3*h);
    a := a + 3*h;
    i := i + 3;
  end;
 Ar := Ar*(3/8)*h;
end;

procedure Resultado;
begin
 gotoxy(31,20);write('Area = ',Ar:4:10);
 gotoxy(21,24);write('Presione cualquier tecla para continuar');
 gotoxy(60,24);c := readkey;
end;

begin
 Inicializa;
 Captura;
 Simpson38;
 Resultado;
end.
```

MÉTODO DE ROMBERG

Este método tiene aplicaciones muy variadas debido a que aúna a la extrapolación de Richardson, la regla del trapecio. La regla extendida del trapecio es de uso sencillo para un gran número de funciones, obteniendo una alta precisión. Así, la regla del trapecio extendida se utilizará para aproximar la integral

$$\int_a^b f(x)\,dx$$

utilizando m subintervalos puede ser escrita como

$$\int_a^b f(x)\,dx = \frac{h}{2}\left[f(a) + f(b) + 2\sum_{j=1}^{m-1} f(x_j)\right] - \left(\frac{b-a}{12}\right)h^2 f''(\mu)$$

donde $a < \mu < b$, $h = (b-a)/m$ y $x_j = a + jh$ para cada $j = 0, 1, 2, \ldots, m$. Ahora obtendremos las aproximaciones preliminares con $m_1 = 1$, $m_2 = 2$, $m_3 = 4$, ..., $m_n = 2^{n-1}$, con n entero y positivo.

$$h_1 = \frac{b-a}{m_1}$$

$$h_1 = \frac{b-a}{2^{n-1}}$$

substituyendo en la regla del trapecio, obtenemos

$$\int_a^b f(x)\,dx = \frac{h_k}{2}\left[f(a) + f(b) + 2\left(\sum_{i=1}^{n-1} f(a + ih_k)\right)\right] - \left(\frac{b-a}{12}\right)h_k^2 f''(\mu_k)$$

donde μ_k es un número dentro del intervalo [a,b].

Denominando $R_{k,1}$ a las aproximaciones preliminares obtenidas con la regla del trapecio y en forma general,

$$R_{k,1} = \frac{1}{2}\left[R_{k-1,1} + h_{k-1}\sum_{i=1}^{r} f\left[a + \left(i - \frac{1}{2}\right)h_{k-1}\right]\right]$$

Para acelerar el proceso se aplica la técnica de extrapolación de Richardson, si la función es continuamente diferenciable cuatro veces en el intervalo [a,b], entonces la regla del trapecio puede ser escrita en la forma

$$\int_a^b f(x)\,dx = \frac{h_k}{2}\left[f(a) + f(b) + 2\left(\sum_{i=1}^{n-1} f(a + ih_k)\right)\right] + \frac{h_k^2}{2}\left[f'(b) - f'(a)\right] + \left(\frac{b-a}{720}\right)h_k^4 f^{(4)}(\mu_k)$$

para cada k = 1, 2, ..., n y alguna de $a < \mu_k < b$.

Con lo cual podemos eliminar el término h_k^2 combinando las ecuaciones, para obtener una fórmula la cual es la aproximación dada por la regla compuesta de Simpson con $h = h_k$.

$$\int_a^b f(x)\,dx = \frac{4R_{k,1} - R_{k-1,1}}{3}$$

para cada k = 2, 3, ..., n y aplicamos la extrapolación a estos valores. En general, si f es

continuamente derivable dos n más dos veces en el intervalo [a,b], entonces para cada k = 1, 2, ..., n.

La regla del trapecio se puede expresar como

$$\int_a^b f(x)\,dx = \frac{h_1}{2}\left[f(a) + f(b) + 2\left[\sum_{i=1}^{2^{k-1}-1} f(a + ih_1)\right]\right]$$

y finalmente obtenemos

$$R_{i,j} = \frac{4^{j-1}R_{i,j-1} - R_{i-1,j-1}}{4^{j-1} - 1}$$

para cada i = 2, 3, 4, ..., n y j = 2, ..., i.

Algoritmo:

Para aproximar la integral $I = \int_a^b f(x)dx$ dados tanto el límite inferior como el superior de la integral y n número de subintervalos en que se divide el intervalo:

Entrada: a, b límites inferior y superior, n número de subintervalos.

Salida: $R_{n,n}$ aproximación a la integral.

Paso 1 $h = a - b$

Paso 2 $R_{1,1} = h*(f(a) - f(b))/a$

Paso 3 para i = 1, ..., n hacer 4 y 5

Paso 4

$$R_{2,1} = \frac{1}{2}\left[R_{1,1} - h\sum_{k=1}^{r} f\left(a + \left(k - \frac{1}{2}\right)h\right)\right]$$

Paso 5 para j = 2, ..., i

$$R_{2,j} = \frac{4^{j-1} R_{2,j-1} - R_{1,j-1}}{4^{j-1} - 1}$$

Paso 6 Si $|R_{2,i} - R_{2,i}| < \varepsilon$ entonces Paso 9

Paso 7 h = h/2

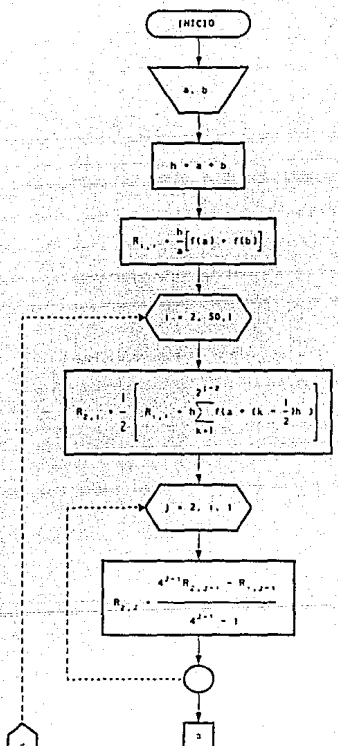
Paso 8 para j = 1, ..., i

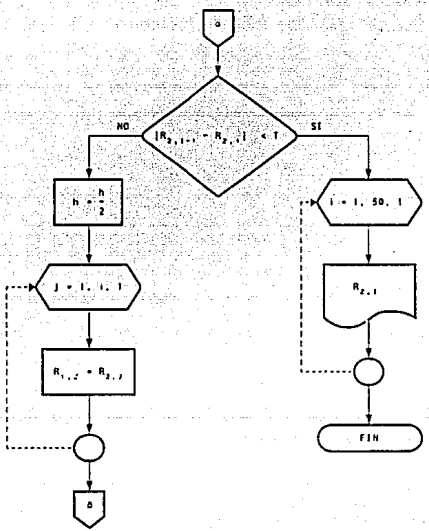
$R_{1,j} = R_{2,j}$

Paso 9 Salida (para i = 1, ..., n, $R_{2,i}$)

Terminar.

METODO DE ROMBERG




a
NO
SI
$|R_{2,i+1} - R_{2,i}| < T$
$h = \frac{h}{2}$
$j = 1, i, 1$
$R_{1,j} = R_{2,j}$
a
$i = 1, 50, 1$
$R_{2,i}$
FIN

```
Program Integracion_de_Romberg;
uses Crt;
const t = 1E-10;
var R : array [1..2,1..50] of real;
   a, b, h, x, Integral : real;
   i, j, k, Dosalaimenos2, Cuatroalajmenos1 : integer;
   mensaje : byte;
   c : char;

function F(X : real) : real;
begin
  F := 1/X;
end;

procedure Inicializa;
begin
  a := 0;
  b := 0;
  h := 0;
  i := 0;
  j := 0;
  k := 0;
  x := 0;
  Integral := 0;
  Dosalaimenos2 := 0;
  Cuatroalajmenos1 := 0;
  mensaje := 0;
  c := char(0);
  for i := 1 to 2 do
    for j := 1 to 50 do
      R[i,j] := 0;
end;

procedure Captura;
begin
  repeat;
  clrscr;
  gotoxy(29,3);writet('INTEGRACION DE ROMBERG');
  gotoxy(25,7);writet('Función a integrar: X^2 - 4x -2');
  gotoxy(25,12);writet('Suministre los siguientes datos');
  gotoxy(22,16);write('Límite inferior de integración: a = ');
  gotoxy(22,18);write('Límite superior de integración: b = ');
  gotoxy(21,24);write('Presione cualquier tecla para continuar');
  gotoxy(58,16);read(a);
  gotoxy(58,18);read(b);
  gotoxy(60,24);c := readkey;
  if a = b then
    begin
      gotoxy(17,13);writet('Los límites de integración deben ser diferentes');
    end;
```

```
  until a <> b;
  clrscr;
  gotoxy(29,5);write('INTEGRACION DE ROMBERG');
  gotoxy(35,12);Textbackground(7);Textcolor(0);
  gotoxy(35,12);write('En  Proceso');
  gotoxy(45,12);Textbackground(0);Textcolor(7);
  gotoxy(32,24);write('Espere un momento');
end;

procedure Inicio;
begin
  i := 1;
  dosalaimenos2 := 1;
  h := b - a;
  R[1,1] := (h/2)*(F(a) + F(b));
end;

procedure Trapecio;
var sigma : real;
begin
  sigma := 0;
  for k := 1 to dosalaimenos2 do
    begin
      sigma := sigma + F(a + (k - 0.5) * h);
    end;
  R[2,1] := (R[1,1] + h*sigma)/2;
end;

procedure Richardson;
begin
  Cuatroalajmenos1 := 1;
  for j := 2 to i do
    begin
      Cuatroalajmenos1 := Cuatroalajmenos1 * 4;
      R[2,j] := (Cuatroalajmenos1*R[2,j-1] - R[1,j-1])
                /(Cuatroalajmenos1 - 1);
    end;
end;

procedure Romberg;
begin
  Inicio;
  for i:= 2 to 50 do
   begin
    Trapecio;
    dosalaimenos2 := dosalaimenos2 * 2;
    Richardson;
```

```
    for j := 1 to i do
      begin
        if (ABS(R[2,j] - R[2,j+1])) <= t then
                                        begin
                                          mensaje := 1;
                                          Integral := R[2,j];
                                          exit;
                                        end;
      end;
    h := h/2;
    for j := 1 to i do
      begin
        R[1,j] := R[2,j];
      end;
    end;
  end;

  procedure Resultado;
  begin
    clrscr;
    if mensaje = 1 then
      begin
        gotoxy(29,5);write('INTEGRACION DE ROMBERG');
        gotoxy(27,8);write('Función evaluada:  Ln 137.2');
        gotoxy(24,12);write('Entre los límites de integración');
        gotoxy(30,14);write('inferior  a = ',a:4:4);
        gotoxy(30,16);write('superior  b = ',b:4:4);
        gotoxy(20,20);write('El valor de la integral es = ',Integral:6:10);
        gotoxy(21,24);write('Presione cualquier tecla para continuar');
        gotoxy(60,24);c := readkey;
      end
    else
      begin
        gotoxy(29,5);write('INTEGRACION DE ROMBERG');
        gotoxy(27,8);write('Función evaluada:  Ln 137.2');
        gotoxy(24,12);write('Entre los límites de integración');
        gotoxy(30,14);write('inferior  a = ',a:4:4);
        gotoxy(30,16);write('superior  b = ',b:4:4);
        gotoxy(16,20);write('El método no converge en 50 iteraciones');
        gotoxy(21,24);write('Presione cualquier tecla para continuar');
        gotoxy(60,24);c := readkey;
      end;
  end;

  begin
    Inicializa;
    Captura;
    Romberg;
    Resultado;
  end.
```

CUADRATURA GAUSSIANA

Este método hace uso de un polinomio interpolante en lugar de evaluar la función en algunos puntos. Se utiliza si f(x) puede evaluarse en cualquir punto. Dicho polinomio de interpolación hace uso de los parámetros listados en la tabla I. El número de parámetros dependerá del número de términos de la fórmula. Por ejemplo, una fórmula de tres términos contendrá seis parámetros, los tres valores de x ahora desconocidos y tres ponderaciones, esto corresponde a un polinomio de interpolación de grado 5.

En el caso de la fórmula de dos términos los cuatro parámetros se determinan a continuación

$$\int_{-1}^{1} f(t) = af(t_1) + bf(t_2)$$

mediante el método de coeficientes indeterminados determinamos la fórmula de integración. Se utiliza el intervalo simétrico de integración para simplificar los cálculos, y denominaremos t a la variable. La fórmula será valida para cualquier polinomio de grado tres; por tanto, cumplirá si

$$f(t) = t^3,\ f(t) = t^2,\quad f(t) = t \ \ para \ \ f(t) = 1$$

$$f(t) = t^3: \quad \int_{-1}^{1} t^3 dt = 0 = at_1^3 + bt_2^3$$

$$f(t) = t^2: \quad \int_{-1}^{1} t^2 dt = \frac{2}{3} = at_1^2 + bt_2^2$$

$$f(t) = t: \quad \int_{-1}^{1} t dt = 0 = at_1 + bt_2$$

$$f(t) = 1: \quad \int_{-1}^{1} dt = 2 = a + b$$

multiplicando la tercera ecuación por t^2_1, y restando a la primera, se tiene

$$0 = 0 + b(t_1^2 - t_2 t_1^2) = b(t_2)(t_2 - t_1)(t_2 + t_1)$$

Se satisface la ecuación anterior bajo cualesquiera de las siguientes condiciones: $b=0$, $t_2=0$, $t_1=t_2$ ó $t_2=-t_1$. Sólo ésta, última posibilidad es satisfactoria. Las otras no son validas ó bien reducen la fórmula a un solo término, de manera que se escoge $t_1=-t_2$. Entonces se encuentra que

$$a = b = 1$$

$$t_2 = -t_1 = \sqrt{\frac{1}{3}} = 0.05773$$

$$\int_{-1}^{1} f(t) = f(-0.5773) + f(0.5773)$$

suponiendo que los límites de integración son de x_0 a x_1, y no de -1 a 1. Para poder utilizar los parámetros tabulados para la cuadratura, se debe cambiar el intervalo de integración a (-1,1) haciendo un cambio de variable que esté relacionado linealmente de acuerdo con la tabla de la siguiente

$$x = \frac{(x_f - x_o)\,t + x_f + x_o}{2}$$

de manera que

$$dx = \frac{x_f - x_o}{2}\,dt$$

entonces

$$\int_{x_o}^{x_f} f(x)\,dx = \frac{x_f - x_o}{2}\int_{-1}^{1} f\left(\frac{(x_f - x_o)\,t - x_f - x_o}{2}\right)dt$$

Tabla I Parámetros de las fórmulas gaussianas

PUNTOS	Z	PONDERACION	VALIDO HASTA GRADO
2	±0.5773502692	1	3
3	±0.7745966692	0.5555555556	5
	0.0000000000	0.8888888889	
4	±0.8611363116	0.3478548451	7
	±0.3399810436	0.6521451549	
5	±0.9061790459	0.2369268851	9
	±0.5384693101	0.4786286705	
	0.0000000000	0.5688888889	
6	±0.9324695142	0.1713244924	

Para obtener los parámetros de las fórmulas gaussianas de orden mayor, se debe resolver el conjunto de ecuaciones que resulta de escribir la definición de f(t), como una sucesión de polinomios. En este caso se empleó la teoría de polinomios ortagonales; los valores de t que satisfacen las ecuaciones, son las raíces de los polinomios de Legendre.

Algoritmo:

Para aproximar la integral $I = \int_a^b f(x)dx$ dados tanto el límite inferior como el superior de la integral y O el orden del método. Los parámetros de las fórmulas se almacenarán dentro del programa:

Entrada: a, b límites inferior y superior, O número de subintervalos.

Salida: area aproximación a la integral.

Paso 1 En caso de O asignar los parámetros correspondientes a el orden del método.

Paso 2 para i = 1, ..., o hacer

$$x_i = \frac{x_o + x_n}{2} + \frac{x_n - x_o}{2} u_i$$

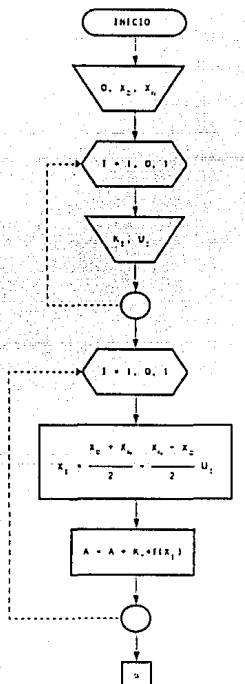
Area = Area + $K_i * f(x_i)$

Paso 3 Area = Area$(x_n - x_o)$

Paso 4 Salida(Area)

Terminar.

METODO DE GAUSS GENERALIZADO



```
Program Gauss_Generalizado;
uses Crt;
var x : array[0..15] of real;
    k, u : array[1..5] of real;
    Xf, A : real;
    i, n : integer;
    c : char;

procedure Inicializa;
begin
  Xf := 0 ;
  A := 0 ;
  i := 0;
  n := 0;
  c := char(0);
end;

function F(X : real) : real;
begin
  F := X*X + 4*X -2;
end;

procedure Captura;
begin
  clrscr;
  gotoxy(28,4);write('METODO GAUSS GENERALIZADO');
  gotoxy(21,7);
  write('Función a integrar: F(x) = x^2 + 4x - 2');
  gotoxy(26,10);write('Suministre los siguientes datos');
  gotoxy(29,12);write('Valor inicial:    a = ');
  gotoxy(29,14);write('Valor final:      b = ');
  gotoxy(29,16);write('Orden del método: n = ');
  gotoxy(21,24);write('Presione cualquier tecla para continuar');
  gotoxy(51,12);read(x[0]);
  gotoxy(51,14);read(XF);
end;

procedure GaussGen;
begin
  gotoxy(51,16);read(n);
  if not((n >= 1) and (n <= 5)) then GaussGen;
  gotoxy(60,24);c := readkey;
  case n of
    1: begin
         k[1] := 1; u[1] := 0;
       end;
    2: begin
         k[1] := 0.5; u[1] := -SQRT(1/3);
         k[2] := 0.5; u[2] :=  SQRT(1/3);
       end;
```

```
  3: begin
       k[1] := 5/18; u[1] := -SQRT(3/5);
       k[2] := 4/9 ; u[2] := 0;
       k[3] := 5/18; u[3] :=  SQRT(3/5);
     end;
  4: begin
       k[1] := 0.1739; u[1] := -0.8611;
       k[2] := 0.3261; u[2] := -0.34;
       k[3] := 0.1739; u[3] :=  0.34;
       k[4] := 0.3261; u[4] :=  0.8211;
     end;
  5: begin
       k[1] := 0.118463; u[1] := -0.906180;
       k[2] := 0.239314; u[2] := -0.538469;
       k[3] := 0.284444; u[3] :=  0;
       k[4] := 0.239314; u[4] :=  0.538469;
       k[5] := 0.118463; u[5] :=  0.906180;
     end;
 end;
 for i := 1 to n do
   begin
     x[i] := (x[0] + Xf)/2 + ((Xf - x[0])/2)*u[i];
     A := A + k[i]*F(x[i]);
   end;
 A := A * (Xf - x[0]);
 gotoxy(20,20);write('El valor de la integral es: ',A:5:10);
 gotoxy(21,24);write('Presione cualquier tecla para continuar');
 gotoxy(60,24);c := readkey;
end;

begin
  Inicializa;
  Captura;
  GaussGen;
end.
```

CAPITULO V

ECUACIONES DIFERENCIALES ORDINARIAS

ECUACIONES DIFERENCIALES ORDINARIAS

El comportamiento de muchos procesos físicos, particularmente los dependientes del tiempo (no estacinarios), pueden representarse por medio de ecuaciones diferenciales ordinarias. Los métodos de solución de estas ecuaciones son por tanto de gran importancia para ingenieros y científicos. Aunque las soluciones analíticas de algunas ecuaciones diferenciales importantes son conocidas, un número todavía mayor de ellas no pueden ser resueltas analíticamente. Afortunadamente, en general se puede encontrar la solución numérica de estas últimas. El presente capítulo describe los procesos más importantes de cálculo de ecuaciones diferenciales ordinarias.

En aplicaciones particulares frecuentemente se requiere de una solución tabulada de una ecuación diferencial sobre un intervalo dado. Si encontramos una fórmula explícita para la solución, podremos utilizarla haciendo uso de tablas trigonométricas ó logarítmicas para obtener la solución tabulada. Si la solución es conocida en forma de serie, ella por si misma es una forma explícita de la cual la solución puede ser calculada para alguna aproximación deseada.

Si no se ha encontrado una fórmula explícita, podremos considerar a la ecuación diferencial como la fórmula que nos permita calcular la solución. Una manera de obtener esto sería aplicar el método de integración por partes. En este capítulo como se menciono antes describiremos varios procedimientos alternativos al ya mencionado. Aún cuando se disponga

de una serie. los métodos pueden darnos la aproximación deseada con un ahorro considerable de trabajo, el uso de estos es mucho más sencillo que el de una fórmula explícita para llegar solución.

NOTA: La ecuación diferencial a evaluar será integrada a cuerpo del programa como una función < f(x,y) > en todos los programas en la sección de declaraciones y se substituirá cada vez que se desee evaluar otra ecuación diferencial diferente.

MÉTODO DE TAYLOR

Considerando la ecuación diferencial

$$\frac{dy}{dx} = f(x, y)$$

con la condición inicial $y = y_n$ cuando $x = x_n$. Dar la solución requerida de esta ecuación será

$$y = \varphi(x)$$

Si $x = x_n$ no es un punto singular de la función, esta última podrá ser expandida en serie de Taylor alrededor de este punto. De este modo, con

$$y_n^{(n)} = \varphi^{(n)}(x_n) = \left[\frac{d^n y}{dx^n}\right]_{x=x_n}$$

$$y = y_n + (x - x_n) y_n^{(1)} + \frac{1}{2!}(x - x_n)$$

$$y_{n+1} = y_n + h y_n' + \frac{h^2}{2!} y_n'' + \frac{h^3}{3!} y_n''' + \frac{h^n}{n!} y^{(n)} + R$$

$$y_{n+1} = \sum_{n=0}^{\infty} \frac{y^{(n)}(x_n)}{n!} h^n$$

una serie de potencias en x que converge sobre algún rango $x_n < x < x_f$. Planteado de esta forma se enfatiza el uso de derivadas totales respecto a x. Ahora con la condición inicial

$$y(x_n) = y_n$$

y de la ecuación diferencial misma

$$y'(x_n) = f(x_n, y_n)$$

substituyendo estos valores de $y(x_n)$ y $y'(x_n)$ en la serie de Taylor, obtenemos los primeros

términos de la serie solución. A continuación, diferenciando la ecuación ordinaria, podemos obtener

$$\frac{d^2y}{dx^2} = [f(x,y)] = \frac{\partial}{\partial x}[f(x,y)] + \frac{\partial}{\partial y}[f(x,y)]\frac{dy}{dx}$$

y, usando la notación f = f(x,y) y

$$f_y = \frac{\partial f}{\partial y}$$

resulta

$$\frac{d^2y}{dx^2} = f_x + ff_y$$

Donde los subíndices denotan las derivadas parciales totales. De manera similar, substituyendo este valor de y''(x) en la serie de Taylor, obtendremos el tercer término de la serie solución. Procediendo de la misma manera, diferenciando sucesivamente para obtener

$$\frac{d^3y}{dx^3}, \quad \frac{d^4y}{dx^4}, \quad \ldots, \quad \frac{d^ny}{dx^n}, \quad \ldots$$

se obtienen los valores

$$y'''(x_o), \quad y^{(iv)}(x_o), \ \ldots, \ y^{(m)}(x_o), \ \ldots$$

Substituyendo estos valores en la serie de Taylor, podemos obtener el cuarto y sus demás términos de la serie solución, los cuales se substituyen para dar una expansión en serie, en x, y evaluar la solución de un paso, h, a partir de un punto dado x.

Algunos inconvenientes de esta técnica son:

- Algunas veces es difícil calcular las derivadas de la serie solución.
- En ocasiones no existen las derivadas parciales calculadas de manera analítica, en el punto en que serán evaluadas.

MÉTODO DE EULER

Una de las técnicas más simples para aproximar soluciones de ecuaciones diferenciales es conocida como método de Euler o método de las tangentes.

Representando la ecuación a resolver como:

$$y' = f(x,y)$$

$$y(x_o) = y_o$$

Si h es un incremento positivo sobre el eje x entonces, como se ve en la figura de la próxima página, podemos encontrar un punto $(x_1,y_1) = (x_o + h, y_1)$ sobre la tangente en (x_o, y_o) a la curva solución desconocida.

De la ecuación de una recta que pasa por un punto dado se tiene

$$y_o = \frac{y_1 - y_o}{(x_o + h) - x_o}$$

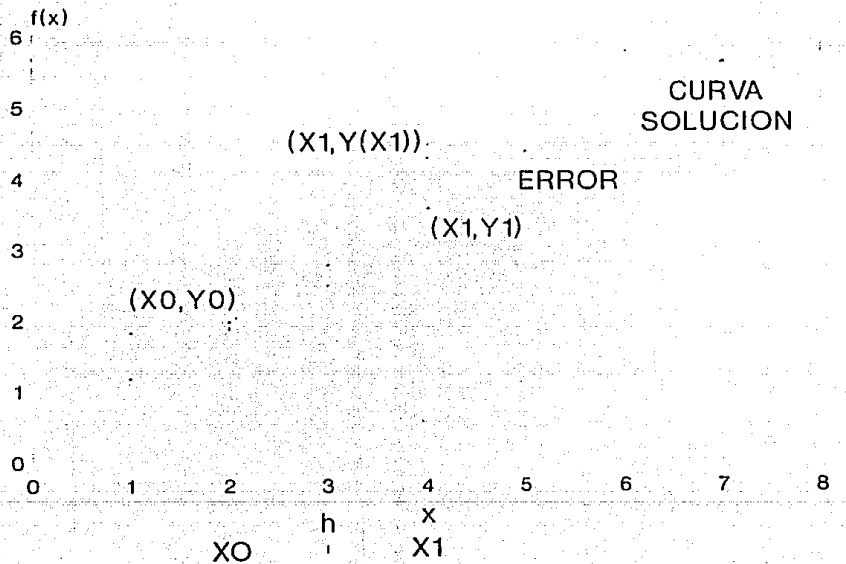
desarrollando, obtenemos

$$y_1 = y_o + hy_o$$

en donde $y'_o + f(x_o,y_o)$, lo substituimos en la ecuación anterior obteniendo

$$y_1 = y_o + hf(x_o,y_o)$$

METODO DE EULER



en general

$$y_{i+1} = y_i + hf(x_i, y_i)$$

donde $x_i = x_o + nh$

La exactitud de la aproximación está en función del tamaño de paso h. Usualmente debemos elegirlo de manera que sea razonablemente pequeño.

La derivación de este método se basa en el uso de la serie de Taylor. Consideraremos $y' = f(x,y)$ expandimos la serie de Taylor hasta el segundo término

$$y(x) = y(0) + y'(0)h$$

la función a resolver es una ecuación diferencial de primer orden con y(x) como solución única

$$y' = f(x,y)$$

con la serie de Taylor hasta el segundo término, y teniendo $h=(x_{n+1} - x_n)$.

$$y(x_{n+1}) = y(x_n) + hy'(x_n)$$

y puesto que y(x) satisface a la ecuación diferencial

$$y(x_{n+1}) = y(x_n) + hf(x_n, y(x_n))$$

Ahora substituimos $y_n = y(x_n)$ para cada i = 1, 2, ..., n, donde $y_o = \alpha$

$$y_{n+1} = y_n + hf(x_n, y_n).$$

Algoritmo:

Para aproximar la solución del problema de valor inicial

$$y' = f(t,y), \quad x_o \leq t \leq x_f, \quad y_o = y(a)$$

Entrada x_o, x_f, y_f puntos extremos, y_o condición inicial, vf variable de control, n número de subintervalos.

Salida y_n, x_n solución aproximada.

Paso 1 Si disponemos de x_f entonces

Paso 1.1 Lee x_f

Paso 1.2 $h = (x_f,y_o)/n$

Paso 1.3 para i =0, ..., n-1

$x_{i+1} = x_i + h$

$y_{i+1} = y_i + f(x_i,y_i)h$, Paso 3

Paso 2 Si disponemos de y_f

Paso 2.1 Lee y_f

Paso 2.2 $h = (x_f,y_o)/n$

Paso 2.3 para i =0, ..., n-1

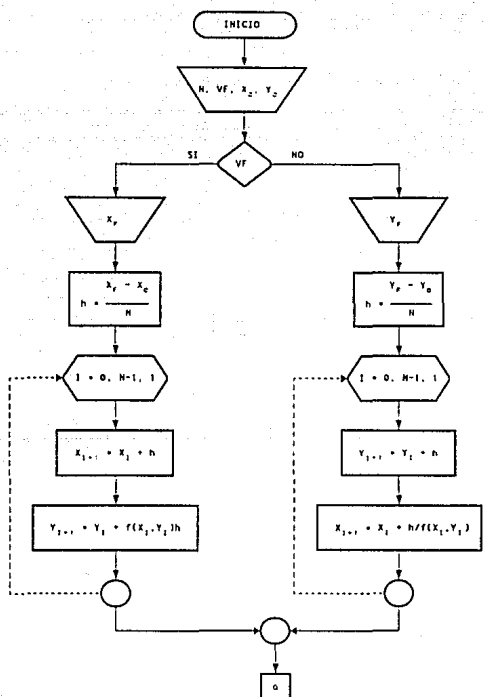
$y_{i+1} = y_i + h$

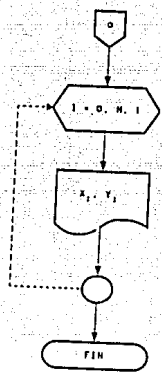
$x_{i+1} = x_i + h/f(x_i,y_i)$, Paso 3

Paso 3 Salida (para i =0, ..., n-1, x_i o y_i)

Terminar.

METODO DE EULER




0
I = 0, N, 1
X_I, Y_I
FIN

```
Program Metodo_de_Euler;
uses Crt;
var X, Y, G : array[0..15] of real;
    Xf, Yf, h : real;
    i, n : integer;
    c : char;
    Op : byte;

function F(x, y : real) : real;
begin
  F := x + 2*x*y;
end;

procedure Inicializa;
begin
  Xf := 0;
  Yf := 0;
  h := 0;
  i := 0;
  n := 0;
  Op := 0;
  c := char(0);
end;

procedure Captura;
begin
  clrscr;
  gotoxy(34,3);write('METODO EULER');
  gotoxy(17,6);
  writeln('Ecuacion diferencial a evaluar: F(x,y) = x +2xy');
  gotoxy(26,9);write('Suministre los siguientes datos');
  gotoxy(30,11);write('Valor inicial: Xo = ');
  gotoxy(30,13);write('Valor inicial: Yo = ');
  gotoxy(24,15);write('N-mero de subintervalos: n = ');
  gotoxy(18,18);
  writeln('Valor final disponible de la ecuacion diferencial');
  gotoxy(37,20);write('1.- Xf');
  gotoxy(37,22);write('2.- Yf');
  gotoxy(35,24);write('Opcion:[_]');
  gotoxy(51,11);readln(x[0]);
  gotoxy(51,13);readln(y[0]);
  gotoxy(53,15);readln(n);
end;

procedure OPXf;
begin
  h := (Xf - X[0])/n;
  for i := 0 to n-1 do
    begin
      x[i+1] := x[i] + h;
```

```
      y[i+1] := y[i] + F(x[i],y[i])*h;
    end;
end;

procedure OPYf;
begin
  h := (Xf - X[0])/n;
  for i := 0 to n-1 do
    begin
      y[i+1] := y[i] + h;
      x[i+1] := x[i] + h / F(x[i],y[i]);
    end;
end;

procedure Euler;
begin
  gotoxy(43,24);read(Op);
  if not((Op <= 2) or (Op >= 1)) then Euler;
  for i := 1 to 3 do
    begin
      gotoxy(1,18+i*2);clreol;
    end;
  case Op of
    1:begin
        gotoxy(31,22);write('Valor final:  Xf = ');
        gotoxy(50,22);readf(Xf);
        gotoxy(20,24);
        write('Presione  cualquier tecla para continuar');
        gotoxy(60,24);c := readkey;
        OPXf;
      end;
    2:begin
        gotoxy(31,22);write('Valor final:  Yf = ');
        gotoxy(50,22);readf(Yf);
        gotoxy(20,24);
        write('Presione  cualquier tecla para continuar');
        gotoxy(60,24);c := readkey;
        OPYf;
      end;
  end;
end;

procedure resultado;
begin
  clrscr;
  gotoxy(34,3);write('METODO EULER');
  gotoxy(17,5);
  write('Ecuaci≤n diferencial evaluada:  F(x,y) = x +2xy');
  gotoxy(13,7);
  write('en el intervalo ',x[0]:5:5,' <= x <= ',Xf:5:5);
```

```
  gotoxy(52,7);
  write(' con h = ',h:5:5);
  if n <= 7 then
   begin
    for i := 0 to n do
    begin
     gotoxy(23,9+i*2);
     write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
    end;
   gotoxy(20,24);
   write('Presione  cualquier tecla para continuar');
   gotoxy(60,24);c := readkey;
  end
  else
   begin
    for i := 0 to 7 do
    begin
     gotoxy(23,9+i*2);
     write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
    end;
    gotoxy(20,24);
    write('Presione  cualquier tecla para continuar');
    gotoxy(60,24);c := readkey;
    for i := 0 to 7 do
    begin
     gotoxy(20,9+i*2);clreol;
    end;
    for i := 8 to n do
     begin
      gotoxy(23,7+(i-7)*2);
      write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
     end;
    gotoxy(20,24);
    write('Presione  cualquier tecla para continuar');
    gotoxy(60,24);c := readkey;
   end;
end;

begin
  Inicializa;
  Captura;
  Euler;
  Resultado;
end.
```

MÉTODO DE EULER MODIFICADO

Este método tiene una mayor exactitud que el método de Euler de un paso, el cual provoca errores de truncación debido a que se utiliza la derivada de la curva aplicada, en cierto modo, delante de donde se está realizando el cálculo. Este método, llamado de Euler modificado o Euler-Gauss, da un método de calculo más económico y más práctico.

Este método es un método predictor corrector: dados dos puntos (x_{n-1}, y_{n-1}) y (x_n, y_n) predecimos el valor del siguiente punto aplicando la fórmula del punto medio a

$$\int_{x_{n-1}}^{x_{n+1}} y'(x)\,dx$$

lo cual nos da

$$z_{n+1} = y_{n-1} + 2hy'_n$$

donde z_{n+1} será el valor predicho de y_{n+1}. Evidentemente utilizando la línea tangente en la mitad del doble intervalo como guía para movernos paralelamente a está. Utilizando el valor pronosticado, después calculamos la pendiente en el punto predicho.

$$z'_{n+1} = f(x_{n+1}, P_{n+1})$$

y aplicando la regla del trapecio a

$$\int_{x_n}^{x_{n+1}} y(x)\,dx$$

lo que da

$$y_{n+1} = y_n + \frac{h}{2}\left(z_{n+1} + y_n\right)$$

como valor corregido y_{n+1}. Aquí usamos el promedio de las pendientes en los extremos de los intervalos de integración, como la pendiente promedio en el intervalo. En otras palabras en este método el "predictor" es el método de Euler. El "corrector" es obtenido con la regla del trapecio, obteniendo el procedimiento general.

$$\begin{aligned} x_{n+1} &= x_n + h \\ z_{n+1} &= y_n + hf(x_n, y_n) \\ y_{n+1} &= y_n + \frac{h}{2}\left[f(x_n, y_n) + f(x_{n+1}, z_{n+1})\right] \end{aligned}$$

Algoritmo:

Para aproximar la solución del problema de valor inicial

$$y' = f(t,y), \quad x_o \leq t \leq x_f, \quad y_o = y(a)$$

Entrada: x_o, x_f, y_f puntos extremos. y_o condición inicial, vf variable de control, n número de subintervalos.

Salida y_n, x_n solución aproximada.

Paso 1 Si disponemos de x_f entonces

Paso 1.1 Lee x_f

Paso 1.2 $h = (x_f, y_o)/n$

Paso 1.3 para $i = 0, \ldots, n-1$

$x_{i+1} = x_i + h$

$G_i = (f(x_i,y_i) + f(x_{i+1},y_i+f(x_i,y_i)h))/2$

$y_{i+1} = y_i + G_i h$, Paso 3

Paso 2 Si disponemos de y_f

Paso 2.1 Lee y_f

Paso 2.2 $h = (x_f, y_o)/n$

Paso 2.3 para $i = 0, \ldots, n-1$

$y_{i+1} = y_i + h$

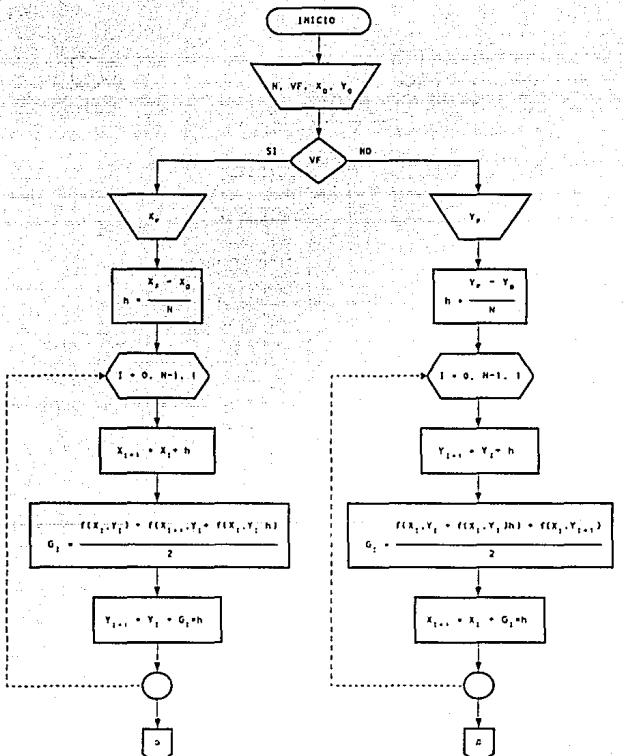
$G_i = (f(x_i,y_i + f(x_i,y_i)h) + f(x_i,y_{i+1}))/2$

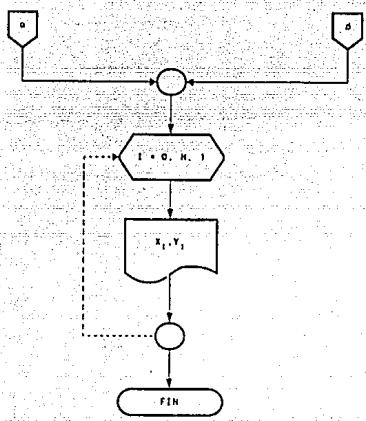
$x_{i+1} = x_i + G_i h$, Paso 3

Paso 3 Salida (para $i = 0, \ldots, n-1$, x_i o y_i)

Terminar.

METODO DE EULER MODIFICADO




D
A
(• O. H.)
X_1, Y_1
FIN

```
Program Metodo_de_Euler_Gauss_o_Euler_modificado;
uses Crt;
var X, Y, G, K : array[0..15] of real;
    Xf, Yf, h : real;
    i, j, n : integer;
    c : char;
    Op : byte;

function F(x, y : real) : real;
begin
  F := x + 2*x*y;
end;

procedure Inicializa;
begin
  Xf := 0;
  Yf := 0;
  h := 0;
  i := 0;
  j := 0;
  n := 0;
  Op := 0;
  c := char(0);
end;

procedure Captura;
begin
  clrscr;
  gotoxy(31,3);write('METODO EULER-GAUSS');
  gotoxy(17,6);write('Ecuación diferencial a evaluar: F(x,y) = x + 2xy');
  gotoxy(26,9);write('Suministre los siguientes datos');
  gotoxy(30,11);write('Valor inicial: Xo = ');
  gotoxy(30,13);write('Valor inicial: Yo = ');
  gotoxy(24,15);write('Número de subintervalos: n = ');
  gotoxy(18,18);
  write('Valor final disponible de la ecuación diferencial');
  gotoxy(37,20);write('1.- Xf = ');
  gotoxy(37,22);write('2.- Yf = ');
  gotoxy(35,24);write('Opcion [_]');
  gotoxy(51,11);read(x[0]);
  gotoxy(51,13);read(y[0]);
  gotoxy(53,15);read(n);
end;
procedure OPXf;
begin
  G[0] := F(x[0],y[0]);
  h := (Xf - X[0])/n;
  for i := 0 to n-1 do
    begin
      x[i+1] := x[i] + h;
```

```
      G[i] := (F(x[i],y[i]) + F(x[i+1],y[i]+F(x[i],y[i])*h))/2;
      y[i+1] := y[i] + G[i]*h;
    end;
end;

procedure OPYf;
begin
  G[0] := i / F(x[0],y[0]);
  h := (Xf - X[0])/n;
  for i := 0 to n-1 do
    begin
      y[i+1] := y[i] + h;
      G[i] := (F(x[i],y[i]) + F(x[i]+F(x[i],y[i])*h,y[i+1]))/2;
      x[i+1] := x[i] + G[i]*h;
    end;
end;

procedure EulerGauss;
begin
  gotoxy(43,24);read(Op);
  if not((Op < = 2) or (Op > = 1)) then EulerGauss;
  for i := 1 to 3 do
    begin
      gotoxy(1,18+i*2);clreol;
    end;
  case Op of
    1:begin
        gotoxy(31,22);write('Valor final:  Xf = ');
        gotoxy(50,22);read(Xf);
        gotoxy(20,24);write('Presione  cualquier tecla para continuar');
        gotoxy(60,24);c := readkey;
        OPXf;
      end;
    2:begin
        gotoxy(31,22);write('Valor final:  Yf = ');
        gotoxy(50,22);read(Yf);
        gotoxy(20,24);write('Presione  cualquier tecla para continuar');
        gotoxy(60,24);c := readkey;
        OPYf;
      end;
  end;
end;

procedure resultado;
begin
  clrscr;
  gotoxy(31,3);write('METODO EULER-GAUSS');
  gotoxy(17,5);write('Ecuación diferencial evaluada: F(x,y) = x + 2xy');
  gotoxy(13,7);write('en el intervalo ',x[0]:5:5,' < = x < = ',Xf:5:5,' con h = ',h:5:5);
```

```
  if n <= 7 then
   begin
    for i := 0 to n do
     begin
      gotoxy(23,9+i*2);write('Punto P',i,' = (',x[i]:5:8,', ',y[i]:5:8,')');
     end;
    gotoxy(20,24);write('Presione cualquier tecla para continuar');
    gotoxy(60,24);c := readkey;
   end
  else
   begin
    for i := 0 to 7 do
    begin
     gotoxy(23,9+i*2);write('Punto P',i,' = (',x[i]:5:8,', ',y[i]:5:8,')');
    end;
    gotoxy(20,24);write('Presione cualquier tecla para continuar');
    gotoxy(60,24);c := readkey;
    for i := 0 to 7 do
    begin
     gotoxy(20,9+i*2);clreol;
    end;
    for i := 8 to n do
     begin
      gotoxy(23,7+(i-7)*2);write('Punto P',i,' = (',x[i]:5:8,', ',y[i]:5:8,')');
     end;
    gotoxy(20,24);write('Presione cualquier tecla para continuar');
    gotoxy(60,24);c := readkey;
   end;
end;

begin
  Inicializa;
  Captura;
  EulerGauss;
  Resultado;
end.
```

MÉTODO DEL PUNTO MEDIO

El método de Euler modificado es referido como un método cerrado a partir de la expresión de aproximación para la siguiente integral

$$y(x_{n+1}) - y(x_{n-1}) = \int_{x_{n-1}}^{x_{n+1}} f(t, y(t))dt = \int_{x_{n-1}}^{x_{n+1}} g(t)dt$$

involucrando $g(x_{n+1})$. La fórmula de cuadratura asociada es cerrada en este caso. Si tal no es el caso, entonces tenemos un método abierto. Por esto el método de Euler el cual esta basado en la regla del rectángulo, tiene una formula de cuadratura abierta, esto es un método abierto. Otro método abierto esta basado en le regla del punto medio de integración, teniendo la expansión anterior podemos representar la integral por

$$\int_{x_{n-1}}^{x_{n+1}} g(t)dt = 2hg(x_n) = 2hf(x_n, y(x_n))$$

la fórmula para el método del punto medio

$$y_{n+1} = y_{n-1} + 2hf(x_n, y_n), \qquad n = 1, 2, \ldots$$

Evidentemente con esta fórmula no podemos encontrar y_1. Uno debe, sin embargo, determinar y_1 por medio de otro procedimiento diferente tal como el método de Euler o el método de Euler modificado.

El método del punto medio es más sencillo de aplicar que el método de Euler pero tiene las siguientes desventajas: primero, el error de la cuadratura, mientras que el del mismo orden en h, es cuatro veces más grande; segundo un procedimiento especial de inicio es requerido; tercero, el método frecuentemente sufre inestabilidad numérica.

En la gráfica de la siguiente página se muestra el error para el caso en que $h < 0$ cuando se utiliza el método de Euler o punto pendiente. Para cualquier solución que crece exponencialmente, la solución calculada siempre se acerca lentamente detrás de la solución exacta. Esto es lo que trata de evitar este método, donde la derivada usada es la correspondiente al punto medio entre las dos abscisas.

Algoritmo:

Para aproximar la solución del problema de valor inicial

$$y' = f(t,y), \quad x_a \leq t \leq x_f, \quad y_a = y(x_a)$$

Entrada: x_a, x_f puntos extremos, y_a condición inicial, n número de subintervalos.

Salida: $f(x_i,y_i)$ solución aproximada.

Paso 1 $h = (x_f - x_a)/n$

Paso 2 para $i = 0, \ldots, n-1$

$k_1 = f(x_i,y_i)$

$k_2 = f(x_i + h/2 \, , \, y_i + hk_1/2)$

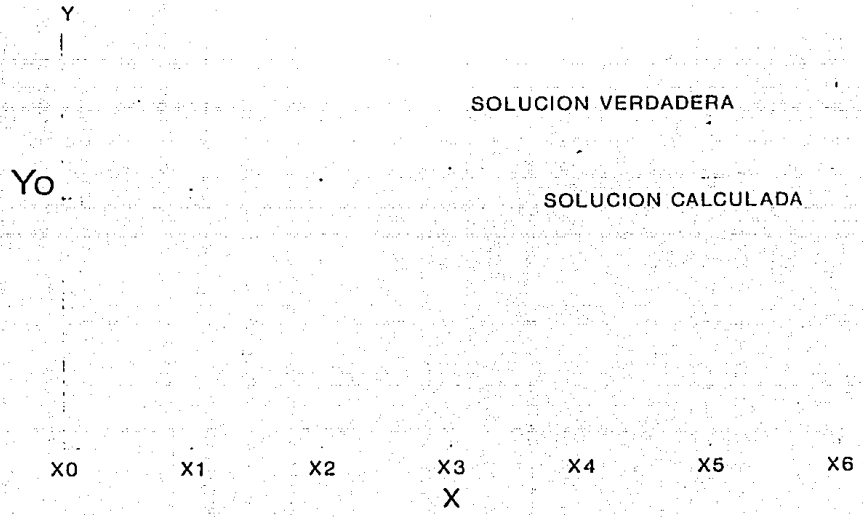
$y_{i+1} = y_i + k_2 h$

$x_{i+1} = x_i + h$

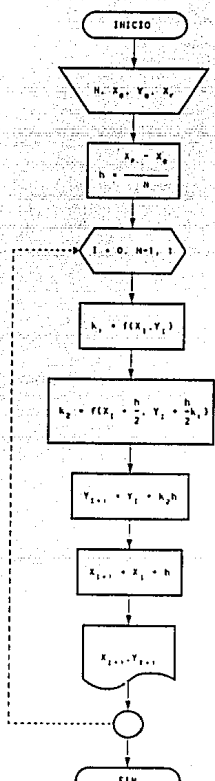
Paso 3 Salida (para $i = 0, \ldots, n-1$, $f(x_i,y_i)$)

Terminar.

PUNTO MEDIO


Y
SOLUCION VERDADERA
Yo
SOLUCION CALCULADA
X0
X1
X2
X3
X4
X5
X6
X

METODO DEL PUNTO MEDIO



```
Program Punto_Medio;
uses Crt;
var x, y : array[0..15] of real;
    xf, h, K1, K2 : real;
    i, n : integer;
    c : char;

function F(x, y : real) : real;
begin
  F := x + 2*x+y;
end;
procedure Inicializa;
begin
  Xf := 0;
  h := 0;
  K1 := 0;
  K2 := 0;
  i := 0;
  n := 0;
  c := char(0);
end;
procedure Captura;
begin
  clrscr;
  gotoxy(28,5);writeln('METODO DEL PUNTO MEDIO');
  gotoxy(17,8);
  write('Ecuacion diferencial a evaluar: F(x,y) = x + 2xy');
  gotoxy(26,11);write('Suministre los siguientes datos');
  gotoxy(30,14);write('Valor inicial:  Xo = ');
  gotoxy(30,16);writeln('Valor inicial:  Yo = ');
  gotoxy(30,18);write('Valor final:    Xf = ');
  gotoxy(24,20);write('Número de subintervalos: n  = ');
  gotoxy(51,14);read(x[0]);
  gotoxy(51,16);read(y[0]);
  gotoxy(51,18);read(Xf);
  gotoxy(53,20);readln(n);
  gotoxy(20,24);write('Presione cualquier tecla para continuar');
  gotoxy(60,24);c := readkey;
end;
procedure PuntoMedio;
begin
  h := (Xf - x[0])/n;
  for i := 0 to n-1 do
    begin
      K1 := F(x[i],y[i]);
      K2 := F(x[i]+h/2,y[i]+h/2*K1);
      y[i+1] := y[i] + K2*h;
      x[i+1] := x[i] + h;
    end;
end;
```

```
procedure Resultado;
begin
  clrscr;
  gotoxy(28,3);write('METODO DEL PUNTO MEDIO');
  gotoxy(17,5);
  write('Ecuación diferencial evaluada:  F(x,y) = x +2xy');
  gotoxy(13,7);
  write('en el intervalo ',x[0]:5:5,' <= x <= ',XI:5:5,
        ' con h = ',h:5:5);
  if n <= 7 then
   begin
    for i := 0 to n do
     begin
      gotoxy(23,9+i*2);
      write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
     end;
    gotoxy(20,24);
    write('Presione  cualquier tecla para continuar');
    gotoxy(60,24);c := readkey;
   end
  else
   begin
    for i := 0 to 7 do
    begin
      gotoxy(23,9+i*2);
      write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
    end;
    gotoxy(20,24);
    write('Presione  cualquier tecla para continuar');
    gotoxy(60,24);c := readkey;
    for i := 0 to 7 do
    begin
      gotoxy(20,9+i*2);clreol;
    end;
    for i := 8 to n do
     begin
       gotoxy(23,7+(i-7)*2);
       write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
     end;
    gotoxy(20,24);
    write('Presione  cualquier tecla para continuar');
    gotoxy(60,24);c := readkey;
   end;
end;
begin
  Inicializa;
  Captura;
  PuntoMedio;
  Resultado;
end.
```

MÉTODO DE HEUN

Método de Heun o método de Euler mejorado, se utiliza cuando se requiere de una aproximación con gran precisión a la solución de una ecuación diferencial ordinaria.

Este método calcula dos derivadas en el intervalo, una en el punto inicial y otra en el punto final, para luego promediarse obteniendo una aproximación mejorada de la pendiente en el intervalo completo. Esto se muestra en la gráfica de la siguiente página.

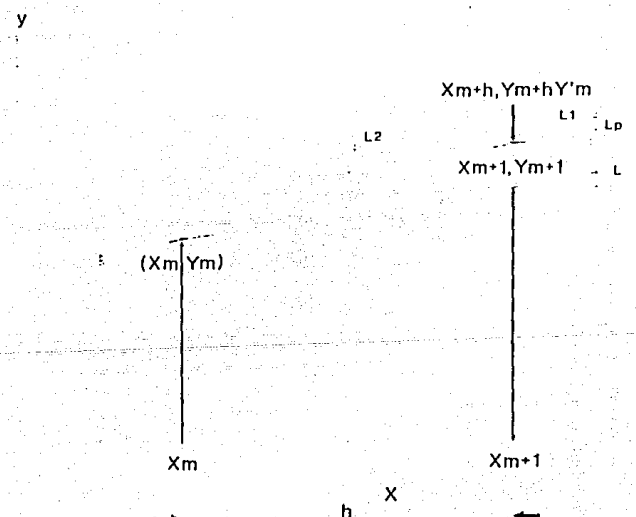
Usamos el método de Euler para determinar el punto x_{m+h}, $y_m+hy'_m$, que esta en la línea L_1 del esquema. En este punto calculamos la pendiente de la curva, lo que nos lleva a la línea L_2. Promediamos las dos pendientes y obtenemos la línea de trazos. Finalmente, dibujamos una línea L, paralela a L_1 y que pasa por el punto x_m, y_m. El punto en que esta línea intersecta ala ordenada levantada en $x = x_{m+1} = x_{m+h}$ se considera el punto x_{m+1}, y_{m+1}. La pendiente de la línea L, que es también la de la línea L_1, es

$$\phi(x_m, y_m, h) = \frac{1}{2}\left[f(x_m, y_m) + f(x_m+h, y_m+hy'_m)\right]$$

en donde

$$y'_m = f(x_m, y_m)$$

METODO DE HEUN



La ecuación de L es entonces

$$y = y_m + (x - x_m)\phi(x_m, y_m, h)$$

así que

$$y_{m+1} = y_m + h\phi(x_m, y_m, h)$$

Las dos últimas ecuaciones definen el método de Heun. Este método tiene un esquema predictor-corrector que queda de la siguiente manera al rearreglar las ecuaciones anteriores

Predictor

$$y^0_{i+1} = y_i + f(x_i, y_i)h$$

Corrector

$$y_{i+1} = y_i + \frac{h}{2}\left[f(x_i, y_i) + f(x_{i+1}, y^0_{i+1})\right]$$

Algoritmo:

Para aproximar la solución del problema de valor inicial

$$y' = f(t,y), \quad x_o \leq t \leq x_f, \quad y_o = y(x_o)$$

Entrada: x_o, x_f puntos extremos, y_o condición inicial, n número de subintervalos.

Salida: $f(x_i,y_i)$ solución aproximada.

Paso 1 $h = (x_f - x_o)/n$

Paso 2 para i = 0, ..., n-1

$k_1 = f(x_i,y_i)$

$k_2 = f(x_i + 3h/2 , y_i + 3hk_1/2)$

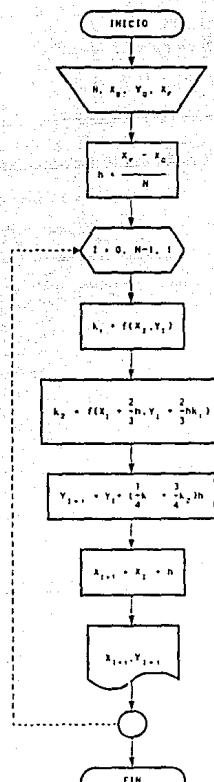
$y_{i+1} = y_i + (k_1/4 + 3k_2/4)h$

$x_{i+1} = x_i + h$

Paso 3 Salida (para i =0, ..., n-1, $f(x_i,y_i)$)

Terminar.

METODO DE HEUN



```
Program Metodo_de_Heun;
uses Crt;
var x, y : array[0..15] of real;
    xf, h, K1, K2 : real;
    i, n : integer;
    c : char;

function F(x, y:: real) : real;
begin
  F := x + 2*x*y;
end;

procedure Inicializa;
begin
  Xf := 0;
  h := 0;
  K1 := 0;
  K2 := 0;
  i := 0;
  n := 0;
  c := char(0);
end;

procedure Captura;
begin
  clrscr;
  gotoxy(33,5);write('METODO DE HEUN');
  gotoxy(17,8);
  write('Ecuación diferencial a evaluar: F(x,y) = x + 2xy');
  gotoxy(26,11);write('Suministre los siguientes datos');
  gotoxy(30,14);write('Valor inicial:  Xo =');
  gotoxy(30,16);write('Valor inicial:  Yo =');
  gotoxy(30,18);write('Valor final:    Xf =');
  gotoxy(24,20);write('Número de subintervalos: n = ');
  gotoxy(51,14);read(x[0]);
  gotoxy(51,16);read(y[0]);
  gotoxy(51,18);read(Xf);
  gotoxy(53,20);read(n);
  gotoxy(20,24);
  write('Presione  cualquier tecla para continuar');
  gotoxy(60,24);c := readkey;
end;

procedure Heun;
begin
  h := (Xf - x[0])/n;
  for i := 0 to n-1 do
    begin
      K1 := F(x[i],y[i]);
      K2 := F(x[i]+(2*h)/3,y[i]+(2*h*k1)/3);
```

```
      y[i+1] := y[i] + (K1/4 + (3*K2)/4)*h;
      x[i+1] := x[i] + h;
    end;
end;

procedure Resultado;
begin
  clrscr;
  gotoxy(33,3);write('METODO DE HEUN');
  gotoxy(17,5);write('Ecuación diferencial evaluada:  F(x,y) = x +2xy');
  gotoxy(13,7);write('en el intervalo ',x[0]:5:5,' <= x <= ',XI:5:5,' con h = ',h:5:5);
  if n <= 7 then
    begin
     for i := 0 to n do
      begin
       gotoxy(23,9+i*2);write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
      end;
     gotoxy(20,24);write('Presione  cualquier tecla para continuar');
     gotoxy(60,24);c := readkey;
    end
   else
     begin
      for i := 0 to 7 do
      begin
        gotoxy(23,9+i*2);write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
      end;
      gotoxy(20,24);write('Presione  cualquier tecla para continuar');
      gotoxy(60,24);c := readkey;
      for i := 0 to 7 do
      begin
        gotoxy(20,9+i*2);clreol;
      end;
      for i := 8 to n do
        begin
          gotoxy(23,7+(i-7)*2);write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
        end;
      gotoxy(20,24);write('Presione  cualquier tecla para continuar');
      gotoxy(60,24);c := readkey;
     end;
end;

begin
  Inicializa;
  Captura;
  Heun;
  Resultado;
end.
```

MÉTODO DE RUNGE-KUTTA

La solución de una ecuación diferencial por desarrollo directo de Taylor de la función objeto no es en general un método práctico si se tienen derivadas de orden mayor que uno. Afortunadamente, es posible deducir algoritmos de paso sencillo que incluyen sólo cálculos de derivadas de primer orden, pero que a su vez producen resultados equivalentes en exactitud a las fórmulas de Taylor de orden superior. Estos algoritmos son los llamados métodos de Runge-Kutta. Las aproximaciones de segundo, tercer y cuarto orden (es decir, aproximaciones con exactitud equivalente al desarrollo de Taylor de y(x) que contenga términos en h^2, h^3, h^4 respectivamente) requieren el cálculo de f(x,y) en dos, tres y cuatro puntos, respectivamente, de x en el intervalo $x_i < x < x_{i+1}$. Los métodos de orden m, siendo m mayor a cuatro, exigen cálculos de las derivadas en más de m puntos.

Todos los método de Runge-Kutta tienen algoritmos de la forma

$$y_{i+1} = y_i + h\phi(x_i, y_i, h)$$

donde ϕ es la función incremento o, más fácilmente, una aproximación de f(x,y) en el intervalo $x_i <= x <= x_{i+1}$ convenientemente elegida.

MÉTODO DE RUNGE-KUTTA DE SEGUNDO ORDEN

Siendo ϕ una medida ponderada de dos valores k1 y k2 de la derivada en el intervalo $x_i <= x <= x_{i+1}$, es decir,

$$\phi = ak1 + bk2$$

el algoritmo de Ruge-Kutta es

$$y_{i+1} = y_i + h(ak1 + bk2) \qquad (0)$$

y

$$k1 = f(x_i, y_i) \qquad (1)$$

$$k2 = f(x_i + ph\ ,\ y_i + phf(x_i, y_i))$$

$$k2 = f(x_i + ph\ ,\ y_i + qhk_1)$$

donde a, b, n, p son coeficientes para que el método de Runge-Kutta reproduzca la serie de Taylor hasta el término h^2. Desarrollamos en primer lugar k2 en serie de Taylor de dos variables, despreciando términos cuyo exponente h es mayor que uno:

$$k2 = f(x_i + ph\ ,\ y_i + qhf(x_i, y_i))$$

$$k2 = f(x_i, y_i) + phf'_x(x_i, y_i) + qhf(x_i, y_i)f'_y(x_i, y_i) + O(h^2) \qquad (2)$$

De las expresiones anteriores (1) y (2) obtenemos

$$y_{i+1} = y_i + h[af(x_i, y_i) + bf(x_i, y_i)]$$

$$+ h^2[bpf'_x(x_i, y_i) + bpf(x_i, y_i)f'_y(x_i, y_i)] + O(h^3) \qquad (3)$$

desarrollando para la función y(x) en x_i por medio de la serie de Taylor

$$y(x_i+h) = y(x_{i+1}) = y(x_i) + hf(x_i, y(x_i)) \\ \frac{h^2}{2!}f'(x_i, y(x_i)) + \frac{h^3}{3!}f''(\xi, y(\xi)), \quad \xi \text{ en } (x_i, x_{i+1}) \qquad (4)$$

empleando la regla de diferenciación de funciones implícitas.

$$\frac{df}{dx} = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx}$$

la derivada viene dada por

$$f'(x_i, y(x_i)) = f_x(x_i, y(x_i)) + f_y(x_i, y(x_i))f(x_i, y(x_i))$$

ahora igualamos potencias de h en (3) y (4). Suponemos $y_i = y(x_i)$, e igualando los coeficientes de h^2 se obtendrá para todas las funciones f(x,y) que cumplan con las condiciones necesarias de diferenciabilidad, a + b = 1 y bp = ½. Por tanto

$$a = 1 - b$$

$$p = q = 1/(2b)$$

Este sistema tiene tres ecuaciones y cuatro incógnitas y, por tanto es indeterminado. Sin embargo el coeficiente b se puede escoger convenientemente, los valores que se suelen tomarse son b = ½ y b = 1.

En el primer caso b = ½, a = ½, p = q = 1, la ecuación (0) resulta

$$y_{i+1} = y_i + ½[f(x_i, y_i) + f(x_i + h, y_i + hf(x_i, y_i))]h \qquad (5)$$

Este algoritmo de paso sencillo se conoce con el nombre de Euler mejorado o método de Heun.

En el segundo caso b = 1, a = 0, p = q = ½, la ecuación obtenida es

$$y_{i+1} = y_i + hf\left(x_i + \frac{h}{2}, y_i + \frac{h}{2} f(x_i, y_i)\right) \qquad (6)$$

este otro algoritmo se conoce con el nombre de método mejorado del polígono o método de Euler modificado. En este método se utiliza nuevamente el método de Euler dos veces sucesivas, obteniendo una aproximación $y_i + (h/2)f(x_i,y_i)$ en el punto medio de $x_i + h/2$. A continuación la ecuación (6) calcula el valor de $f(x,y)$ para $x = x_i + h/2$, $y = y_i + (h/2)f(x_i,y_i)$, y utiliza este valor medio de la derivada para la totalidad del intervalo (método del punto medio).

MÉTODO DE RUNGE-KUTTA DE TERCER ORDEN

La función incremento para el método de tercer orden es

$$\phi = ak1 + bk2 + ck3$$

siendo k1, k2 y k3 aproximaciones a la derivada en varios puntos del intervalo de integración $[x_i, x_{i+1}]$. En este caso se tiene

$$k1 = f(x_i, y_i)$$

$$k2 = f(x_i + ph, y_i + phk_1)$$

$$k3 = f(x_i + kh, y_i + shk2 + (r-s)hk1)$$

Los algoritmos de Ruge-Kutta de tercer orden vienen dados por

$$y_{i+1} = y_i + h(ak1 + bk2 + ck3)$$

El cálculo de los coeficientes a, b, c, p, r y s se realizan desarrollando k2 y k3 sobre (x_i, y_i) en serie de Taylor como función de dos variables. A continuación se desarrolla la función objeto y(x) también en serie de Taylor como anteriormente, y se igualan los coeficientes de las potencias de h hasta h^3, obteniendo una fórmula con error de truncamiento total de h^4. Los detalles son esencialmente los mismo que para el desarrollo de los métodos de segundo orden

Nuevamente aparecen menos ecuaciones que incógnitas

$$a + b + c = 1$$

$$bp + cr = 1/2$$

$$bp^2 + cr^2 = 1/3$$

$$cps = 1/6$$

Dos de los coeficientes a, b, c, p, r, y s tienen un valor adecuado. Para un conjunto de valores elegido por Kutta, el método de tercer orden es,

$$y_{i+1} = y_i + \frac{h}{6}(k1 + 4k2 + k3)$$

$$k1 = f(x_i, y_i)$$

$$k2 = f\left(x_i + \frac{1}{2}h,\ y_i + \frac{1}{2}hk1\right)$$

$$k3 = f(x_i + h,\ y_i + 2hk2 - hk1)$$

Note que si $f(x,y)$ es función únicamente de x, la fórmula anterior se reduce a la regla de Simpson.

Algoritmo **(Runge-Kutta 3er. orden y 5 constantes)**

Para aproximar la solución del problema de valor inicial

$$y' = f(t,y), \qquad x_o \le t \le x_f, \qquad y_o = y(x_o)$$

Entrada: x_o, x_f puntos extremos, y_o condición inicial, n número de subintervalos.

Salida: $f(x_i,y_i)$ solución aproximada.

Paso 1 $h = (x_f - x_o)/n$

Paso 2 para $i = 0, \ldots, n-1$

$k1 = f(x_i,y_i)$

$k2 = f(x_i + h/2\ ,\ y_i + hk1/2)$

$k3 = f(x_i + 3h/4\ ,\ y_i + 3hk2/4)$

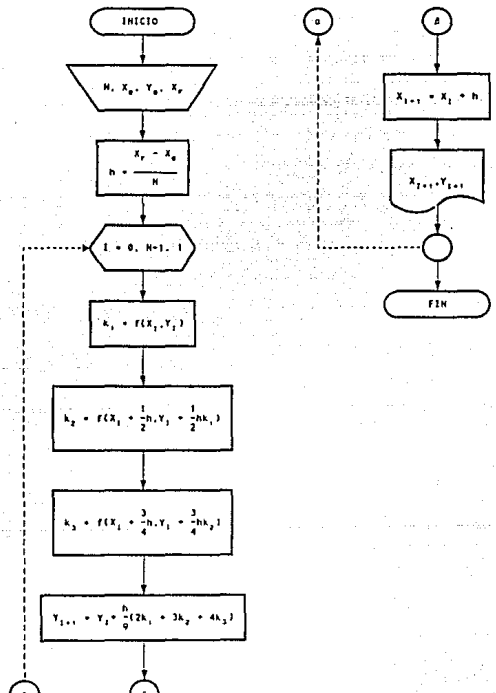
$y_{i+1} = y_i + h(2k1 + 3k2 + 4k3)/9$

$x_{i+1} = x_i + h$

Paso 3 **Salida** (para $i = 0, \ldots, n-1$, $f(x_i,y_i)$)

Terminar.

METODO DE RUNGE-KUTTA 3ER ORDEN Y 5 CONSTANTES



```
Program Metodo_Runge_Kutta_3er_orden_5_constantes;
uses Crt;
var X, Y : array[0..15] of real;
    Xf, h, k1, k2, k3 : real;
    i, n : integer;
    c : char;

function F(x, y : real) : real;
begin
  F := x + 2*x*y;
end;

procedure Inicializa;
begin
  Xf := 0;
  k1 := 0;
  k2 := 0;
  k3 := 0;
  h := 0;
  i := 0;
  n := 0;
  c := char(0);
end;

procedure Captura;
begin
  clrscr;
  gotoxy(31,3);writeln('METODO RUNGE-KUTTA');
  gotoxy(29,5);writeln('3er orden y 5 constantes');
  gotoxy(17,8);
  writeln('Ecuación diferencial a evaluar: F(x,y) = x + 2xy');
  gotoxy(26,11);writeln('Suministre los siguientes datos');
  gotoxy(30,14);writeln('Valor inicial:  Xo = ');
  gotoxy(30,16);writeln('Valor inicial:  Yo = ');
  gotoxy(30,18);writeln('Valor final:    Xf = ');
  gotoxy(27,20);writeln('Número de incrementos: n = ');
  gotoxy(20,24);
  write('Presione cualquier tecla para continuar');
  gotoxy(51,14);readln(x[0]);
  gotoxy(51,16);readln(y[0]);
  gotoxy(51,18);readln(Xf);
  gotoxy(54,20);readln(n);
  gotoxy(60,24);c := readkey;
end;
procedure RK35;
begin
  h := (Xf - x[0])/n;
  for i := 0 to n-1 do
    begin
      k1 := F(x[i] , y[i]);
      k2 := F(x[i] + 0.5*h , y[i] + 0.5*h*k1);
      k3 := F(x[i] + 0.75*h , y[i] + 0.75*h*k2);
      y[i+1] := y[i] + h*((2/9)*k1 + (3/9)*k2 + (4/9)*k3);
      x[i+1] := x[i] + h;
    end;
end;
```

```
procedure resultado;
begin
  clrscr;
  gotoxy(31,2);write('METODO  RUNGE-KUTTA');
  gotoxy(29,3);write('3er orden y 5 constantes');
  gotoxy(17,5);
  write('Ecuación diferencial evaluada: F(x,y) = x + 2xy');
  gotoxy(13,7);
  write('en el intervalo ',x[0]:5:5,' <= x <= ',X:5:5,
        ' con h = ',h:5:5);
  if n <= 7 then
   begin
    for i := 0 to n do
     begin
      gotoxy(23,9+i*2);
      write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
     end;
    gotoxy(20,24);
    write('Presione  cualquier tecla para continuar');
    gotoxy(60,24);c := readkey;
   end
  else
    begin
     for i := 0 to 7 do
     begin
       gotoxy(23,9+i*2);
       write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
     end;
     gotoxy(20,24);
     write('Presione  cualquier tecla para continuar');
     gotoxy(60,24);c := readkey;
     for i := 0 to 7 do
     begin
       gotoxy(20,9+i*2);clreol;
     end;
     for i := 8 to n do
       begin
         gotoxy(23,7+(i-7)*2);
         write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
       end;
     gotoxy(20,24);
     write('Presione  cualquier tecla para continuar');
     gotoxy(60,24);c := readkey;
    end;
end;

begin
  Inicializa;
  Captura;
  RK35;
  Resultado;
end.
```

Algoritmo **(Runge-Kutta 3er. orden y 6 parámetros)**

Para aproximar la solución del problema de valor inicial

$$y' = f(t,y), \quad x_o \leq t \leq x_f, \quad y_o = y(x_o)$$

Entrada: x_o, x_f puntos extremos, y_o condición inicial, n número de subintervalos.

Salida: $f(x_i,y_i)$ solución aproximada.

Paso 1 $h = (x_f - x_o)/n$

Paso 2 para i = 0, ..., n-1

$k1 = f(x_i,y_i)$

$k2 = f(x_i + h/2 \ , \ y_i + hk1/2)$

$k3 = f(x_i + h \ , \ y_i + 2hk2 - hk1)$

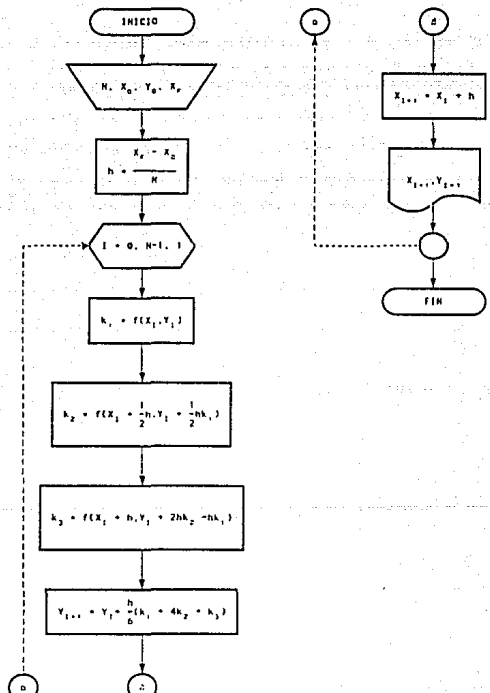
$y_{i+1} = y_i + h(k1 + 4k2 + k3)/6$

$x_{i+1} = x_i + h$

Paso 3 Salida (para i =0, ... n-1. $f(x_i,y_i)$)

Terminar.

METODO DE RUNGE-KUTTA 3ER ORDEN Y 6 PARAMETROS



```
Program Metodo_Runge_Kutta_3er_orden_6_parametros;
uses Crt;
var X, Y : array[0..15] of real;
    Xf, h, k1, k2, k3 : real;
    i, n : integer;
    c : char;

function F(x, y : real) : real;
begin
  F := x + 2*x*y;
end;

procedure Inicializa;
begin
  Xf := 0;
  k1 := 0;
  k2 := 0;
  k3 := 0;
  h := 0;
  i := 0;
  n := 0;
  c := char(0);
end;
procedure Captura;
begin
  clrscr;
  gotoxy(31,3);write('METODO RUNGE-KUTTA');
  gotoxy(28,5);write('3er orden y 6 parámetros');
  gotoxy(17,8);
  write('Ecuacion diferencial a evaluar: F(x,y) = x + 2xy');
  gotoxy(26,11);write('Suministre los siguientes datos');
  gotoxy(30,14);write('Valor inicial:  Xo = ');
  gotoxy(30,16);write('Valor inicial:  Yo = ');
  gotoxy(30,18);write('Valor final:    Xf = ');
  gotoxy(27,20);write('Número de incrementos: n = ');
  gotoxy(20,24);
  write('Presione  cualquier tecla para continuar');
  gotoxy(51,14);read(x[0]);
  gotoxy(51,16);read(y[0]);
  gotoxy(51,18);read(Xf);
  gotoxy(54,20);read(n);
  gotoxy(60,24);c := readkey;
end;

procedure RK36;
begin
  h := (Xf - x[0])/n;
  for i := 0 to n-1 do
    begin
      k1 := F(x[i] , y[i]);
      k2 := F(x[i] + 0.5*h , y[i] + 0.5*h*k1);
      k3 := F(x[i] + h , y[i] + 2*h*k2 - h*k1);
      y[i+1] := y[i] + (h/6)*(k1 + 4*k2 + k3);
      x[i+1] := x[i] + h;
    end;
end;
```

```
procedure resultado;
begin
 clrscr;
 gotoxy(31,2);write('METODO  RUNGE-KUTTA');
 gotoxy(28,3);write('3er orden y 6 parámetros');
 gotoxy(17,5);
 write('Ecuación diferencial evaluada: F(x,y) = x + 2xy');
 gotoxy(13,7);
 write('en el intervalo ',x[0]:5:5,' <= x <= ',Xf:5:5,
       ' con h = ',h:5:5);
 if n <= 7 then
  begin
   for i := 0 to n do
    begin
     gotoxy(23,9+i*2);
     write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
    end;
   gotoxy(20,24);
   write('Presione  cualquier tecla para continuar');
   gotoxy(60,24);c := readkey;
  end
 else
   begin
    for i := 0 to 7 do
    begin
      gotoxy(23,9+i*2);
      write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
    end;
    gotoxy(20,24);
    write('Presione  cualquier tecla para continuar');
    gotoxy(60,24);c := readkey;
    for i := 0 to 7 do
    begin
      gotoxy(20,9+i*2);clreol;
    end;
    for i := 8 to n do
      begin
        gotoxy(23,7+(i-7)*2);
        write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
      end;
    gotoxy(20,24);
    write('Presione  cualquier tecla para continuar');
    gotoxy(60,24);c := readkey;
   end;
end;

begin
 Inicializa;
 Captura;
 RK36;
 Resultado;
end.
```

MÉTODO DE RUGE-KUTTA DE CUARTO ORDEN

Todas las fórmulas de cuarto orden son de la forma siguiente

$$y_{i+1} = y_i + h(ak1 + bk2 + ck3 + dk4)$$

donde k1, k2, k3 y k4 son valores aproximados de la derivada calculados en el intervalo $x_i <= x <= x_{i+1}$. Se utilizan distintos algoritmos de curto orden. Se atribuye a Kutta el siguiente:

$$y_{i+1} = y_i + \frac{h}{6}(k1 + 2k2 + 2k3 + k4)$$

$$k1 = f(x_i, y_i)$$
$$k2 = f\left(x_i + \frac{1}{2}h\ ,\ y_i + \frac{1}{2}hk1\right)$$
$$k3 = f\left(x_i + \frac{1}{2}h\ ,\ y_i + \frac{1}{2}hk2\right)$$
$$k4 = f(x_i + h\ ,\ y_i + hk3)$$

Observe que nuevamente se reduce a la regla de Simpson en caso de que f(x,y) sea únicamente función de x.

Algoritmo:

Para aproximar la solución del problema de valor inicial

$$y' = f(t,y), \quad x_o \le t \le x_t, \quad y_o = y(x_o)$$

Entrada: x_o, x_t puntos extremos, y_o condición inicial, n número de subintervalos.

Salida: $f(x_i,y_i)$ solución aproximada.

Paso 1 $h = (x_t - x_o)/n$

Paso 2 para $i = 0, \ldots, n-1$

$k1 = f(x_i,y_i)$

$k2 = f(x_i + h/2\ ,\ y_i + hk1/2)$

$k3 = f(x_i + h/2\ ,\ y_i + hk2/2)$

$k4 = f(x_i + h\ ,\ y_i + hk3)$

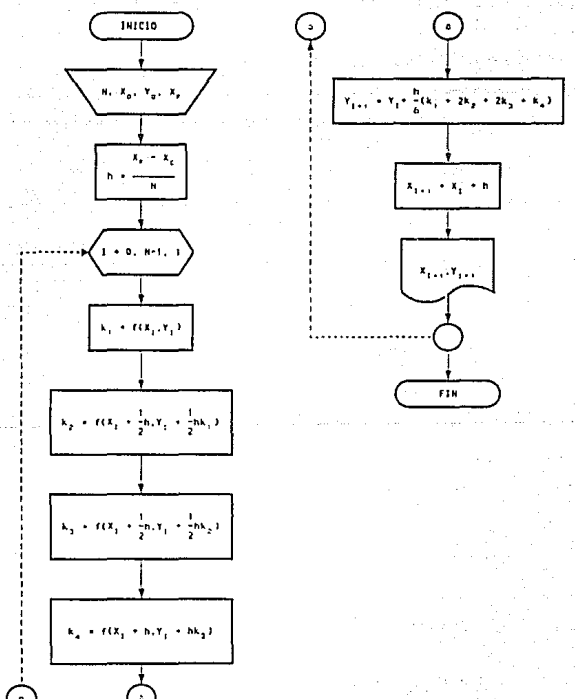
$y_{i+1} = y_i + h(k1 + 2k2 + 2k3 + k4)/6$

$x_{i+1} = x_i + h$

Paso 3 Salida (para $i = 0, \ldots, n-1$. $f(x_i,y_i)$)

Terminar.

METODO DE RUNGE-KUTTA 4º ORDEN Y 7 CONSTANTES



```
Program Metodo_Runge_Kutta_4o_orden_7_constantes;
uses Crt;
var X, Y : array[0..15] of real;
    Xf, h, k1, k2, k3, k4 : real;
    i, n : integer;
    c : char;

function F(x, y : real) : real;
begin
  F := x + 2*x*y;
end;

procedure Inicializa;
begin
  Xf := 0;
  k1 := 0;
  k2 := 0;
  k3 := 0;
  k4 := 0;
  h := 0;
  i := 0;
  n := 0;
  c := char(0);
end;

procedure Captura;
begin
  clrscr;
  gotoxy(31,3);write('METODO  RUNGE-KUTTA');
  gotoxy(29,5);write('4o orden y 7 constantes');
  gotoxy(17,8);
  write('Ecuación diferencial a evaluar: F(x,y) = x + 2xy');
  gotoxy(26,11);write('Suministre los siguientes datos');
  gotoxy(30,14);write('Valor inicial:  Xo = ');
  gotoxy(30,16);write('Valor inicial:  Yo = ');
  gotoxy(30,18);write('Valor final:    Xf = ');
  gotoxy(27,20);write('Número de incrementos: n = ');
  gotoxy(20,24);
  write('Presione  cualquier tecla para continuar');
  gotoxy(51,14);read(x[0]);
  gotoxy(51,16);read(y[0]);
  gotoxy(51,18);read(Xf);
  gotoxy(54,20);read(n);
  gotoxy(60,24);c := readkey;
end;

procedure RK47;
begin
  h := (Xf - x[0])/n;
  for i := 0 to n-1 do
    begin
      k1 := F(x[i] , y[i]);
      k2 := F(x[i] + 0.5*h , y[i] + 0.5*h*k1);
      k3 := F(x[i] + 0.5*h , y[i] + 0.5*h*k2);
      k4 := F(x[i] + h , y[i] + h*k3);
      y[i+1] := y[i] + (h/6)*(k1 + 2*k2 + 2*k3 + k4);
```

```
      x[i+1] := x[i] + h;
    end;
  end;

  procedure resultado;
  begin
    clrscr;
    gotoxy(31,2);write('METODO RUNGE-KUTTA');
    gotoxy(29,3);write('de orden y 7 constantes');
    gotoxy(17,5);
    write('Ecuacion diferencial evaluada: F(x,y) = x + 2xy');
    gotoxy(13,7);
    write('en el intervalo ',x[0]:5:5,' <= x <= ',X:5:5,
          ' con h = ',h:5:5);
    if n <= 7 then
     begin
      for i := 0 to n do
       begin
        gotoxy(23,9+i*2);
        write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
       end;
      gotoxy(20,24);
      write('Presione  cualquier tecla para continuar');
      gotoxy(60,24);c := readkey;
     end
    else
      begin
       for i := 0 to 7 do
       begin
         gotoxy(23,9+i*2);
         write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
       end;
       gotoxy(20,24);
       write('Presione  cualquier tecla para continuar');
       gotoxy(60,24);c := readkey;
       for i := 0 to 7 do
       begin
         gotoxy(20,9+i*2);clreol;
       end;
       for i := 8 to n do
         begin
           gotoxy(23,7+(i-7)*2);
           write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
         end;
       gotoxy(20,24);
       write('Presione  cualquier tecla para continuar');
       gotoxy(60,24);c := readkey;
      end;
  end;

  begin
    Inicializa;
    Captura;
    RK47;
    Resultado;
  end.
```

MÉTODO DE RUNGE-KUTTA DE CUARTO ORDEN Y DIEZ CONSTANTES

Otro de los métodos atribuidos a Kutta es:

$$y_{i+1} = y_i + \frac{h}{8}(k1 + 3k2 + 3k3 + k4)$$

$$k1 = f(x_i, y_i)$$
$$k2 = f\left(x_i + \frac{1}{3}h\ ,\ y_i + \frac{1}{3}hk1\right)$$
$$k3 = f\left(x_i + \frac{2}{3}h\ ,\ y_i - \frac{1}{3}hk1 + hk2\right)$$
$$k4 = f(x_i + h\ ,\ y_i + hk1 - hk2 + hk3)$$

de igual manera este también se reducir a la regla de Simpson si f(x,y) es función únicamente de k.

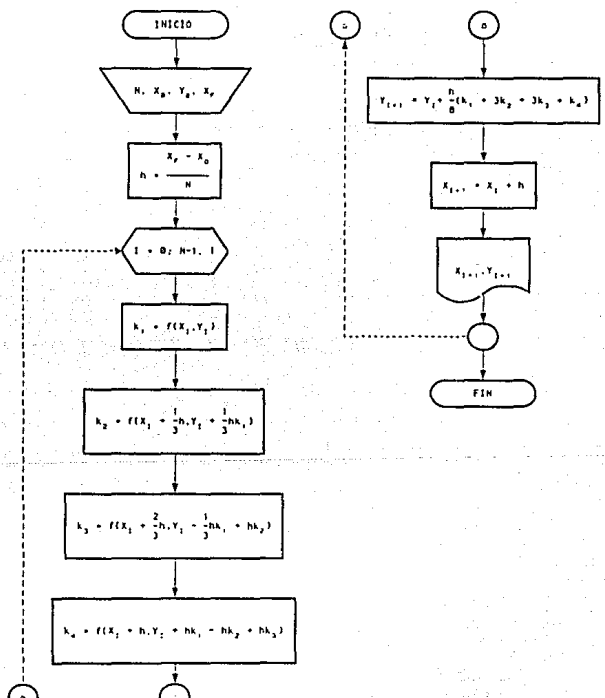
Algoritmo:

Para aproximar la solución del problema de valor inicial

$$y' = f(t,y), \quad x_o \le t \le x_n, \quad y_o = y(x_o)$$

Entrada: x_o, x_n puntos extremos, y_o condición inicial, n número de subintervalos.
Salida: $f(x_i,y_i)$ solución aproximada.
Paso 1 $h = (x_n - x_o)/n$
Paso 2 para i = 0, ..., n-1
k1 = f(x_i, y_i)
k2 = f(x_i + h/3 , y_i + hk1/3)
k3 = f(x_i + 2h/3 , y_i - hk1/3 + hk2)
k4 = f(x_i + h , y_i + hk1 - hk2 + hk3)
$y_{i+1} = y_i + h(k1 + 3k2 + 3k3 + k4)/8$
$x_{i+1} = x_i + h$
Paso 3 Salida (para i =0, ..., n-1, $f(x_i,y_i)$)
Terminar.

METODO DE RUNGE-KUTTA 4° ORDEN Y 10 CONSTANTES



```
Program Metodo_Runge_Kutta_4to_orden_10_constantes;
uses Crt;
var X, Y : array[0..15] of real;
    Xf, h, k1, k2, k3, k4 : real;
    i, n : integer;
    c : char;

function F(x, y : real) : real;
begin
  F := ((1+x)*y*y)/2;
end;

procedure Inicializa;
begin
  Xf := 0;
  k1 := 0;
  k2 := 0;
  k3 := 0;
  k4 := 0;
  h := 0;
  i := 0;
  n := 0;
  c := char(0);
end;

procedure Captura;
begin
  clrscr;
  gotoxy(31,3);write('METODO RUNGE-KUTTA');
  gotoxy(29,5);write('4to orden y 10 constantes');
  gotoxy(17,8);
  write('Ecuación diferencial a evaluar: F(x,y) = x + 2xy');
  gotoxy(26,11);write('Suministre los siguientes datos');
  gotoxy(30,14);write('Valor inicial:  Xo = ');
  gotoxy(30,16);write('Valor inicial:  Yo = ');
  gotoxy(30,18);write('Valor final:    Xf = ');
  gotoxy(27,20);write('Número de incrementos: n = ');
  gotoxy(20,24);
  write('Presione  cualquier tecla para continuar');
  gotoxy(51,14);read(x[0]);
  gotoxy(51,16);read(y[0]);
  gotoxy(51,18);read(Xf);
  gotoxy(54,20);read(n);
  gotoxy(60,24);c := readkey;
end;

procedure RK410;
begin
  h := (Xf - x[0])/n;
  for i := 0 to n-1 do
    begin
      k1 := F(x[i] , y[i]);
      k2 := F(x[i] + h/3 , y[i] + (h*k1)/3);
      k3 := F(x[i] + (2*h)/3 , y[i] - (h*k1)/3 + h*k2);
      k4 := F(x[i] + h , y[i] + h*(k1 - k2 + k3));
      y[i+1] := y[i] + (h/8)*(k1 + 3*k2 + 3*k3 + k4);
```

```
    x[i+1] := x[i] + h;
  end;
end;

procedure resultado;
begin
  clrscr;
  gotoxy(31,2);write('METODO  RUNGE-KUTTA');
  gotoxy(29,3);write('4to orden y 10 constantes');
  gotoxy(17,5);
  write('Ecuación diferencial evaluada: F(x,y) = x + 2xy');
  gotoxy(13,7);
  write('en el intervalo ',x[0]:5:5,' <= x <= ',X:5:5,
        ' con h = ',h:5:5);
  if n <= 7 then
   begin
    for i := 0 to n do
     begin
      gotoxy(23,9+i*2);
      write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
     end;
    gotoxy(20,24);
    Write('Presione  cualquier tecla para continuar');
    gotoxy(60,24);c := readkey;
   end
  else
    begin
     for i := 0 to 7 do
     begin
       gotoxy(23,9+i*2);
       write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
     end;
     gotoxy(20,24);
     write('Presione  cualquier tecla para continuar');
     gotoxy(60,24);c := readkey;
     for i := 0 to 7 do
     begin
       gotoxy(20,9+i*2),clreol;
     end;
     for i := 8 to n do
       begin
         gotoxy(23,7+(i-7)*2);
         write('Punto P',i,' = (',x[i]:5:8,' , ',y[i]:5:8,')');
       end;
     gotoxy(20,24);
     write('Presione  cualquier tecla para continuar');
     gotoxy(60,24);c := readkey;
    end;
end;

begin
  Inicializa;
  Captura;
  RK410;
  Resultado;
end.
```

MÉTODOS MULTIPASO

Los métodos de tipo Runge-Kutta (los cuales incluyen los métodos de Euler, Euler mejorado y Euler modificado como caso especial). Se llaman métodos de un paso debido a que la aproximación involucra solo la información del último paso calculado. Con esto ellos tiene la capacidad de realizar la siguiente fase con un tamaño de paso diferente, y son ideales para iniciar la solución en donde solo se dispone de las condiciones iniciales. Sin embargo, durante la ejecución del método ha comenzado, la información adicional disponible acerca de la función (y sus derivadas), no se retiene para usarla directamente en aproximaciones futuras. Consecuentemente, toda la información utilizada por estos métodos se obtiene dentro del intervalo en el cual se esta aproximando la solución. Un método de pasos múltiples o multipaso es aquel que toma ventaja de este hecho, haciendo uso de la información acerca de la aproximación en más de un punto para determinar las siguientes aproximaciones. Es decir, se apoya en varios puntos anteriores para evaluar el punto siguiente. Estos puntos anteriores se pueden calcular con los métodos de un paso, ya sean los métodos de Euler o los de Runge-Kutta.

MÉTODOS MULTIPASO DE INTEGRACIÓN ABIERTA

Se desarrollan utilizando el polinomio fundamental de Newton, con diferencias finitas sin dividir hacia atrás para expander la función f(x,y); como se vera posteriormente las diferencias sin dividir hacia atrás no afectan el intervalo $[x_n, x_{n-1}]$ y por eso se les denomina métodos de integración abierta.

NOTA: Los algoritmos de los métodos de Adams-Bashforth de 2, 3, 4 y 5 pasos, se integraron en un solo algoritmo, y por lo tanto se presenta un sólo programa para estos métodos.

MÉTODO DE ADAMS-BASHFORTH DE DOS PASOS

Para la derivación de los métodos multipaso utilizaremos la ecuación diferencial

$$y' = f(x,y)$$

en el intervalo $[x_n, x_{n+1}]$ integrando esta ecuación diferencial tendremos

$$\int y\,dx = \int f(x, y(x))\,dx$$

o sea

$$y_{n+1} = y_n + \int f(x, y(x))\,dx \qquad (1)$$

Para realizar la integración aproximamos ahora $f(x,y(x))$ por un polinomio que interpola $f(x,y(x))$ en los $m+1$ puntos $x_n, x_{n-1}, x_{n-2}, \ldots, x_{n-m}$. Utilizando la notación,

$$f_k = f(x_n, y(x_n))$$

podemos utilizar el polinomio de Newton de diferencias hacia atrás de grado m para este propósito

$$P_m(x) = \sum_{k=0}^{m} (-1)^k \binom{-s}{k} \Delta^k f_{n-k}$$

$$s = \frac{x - x_n}{h}$$

Insertando esto en (1) y observando que $d_s = h ds$, obtenemos

$$y_{n+1} = y_n + h\int_0^1 \sum_{k=0}^{m} (-1)^k \binom{-s}{k} \Delta^k f_{n-k} ds \qquad (2)$$

$$y_{n+1} = y_n + h(\gamma_0 f_n - \gamma_1 \Delta f_{n-1} + \ldots + \gamma_m \Delta^m f_{n-m}) \qquad (3)$$

donde

$$\gamma_k = (-1)^k \binom{-s}{k} ds$$

a partir de la función binomial se calculan fácilmente los γ_k. Los primeros de los cuales son

$$\gamma_0 = 1, \quad \gamma_1 = \frac{1}{2}, \quad \gamma_2 = \frac{5}{12}, \quad \gamma_3 = \frac{3}{8}, \quad \gamma_4 = \frac{251}{720}$$

para el método de dos pasos la ecuación (2) tiene como valor de m = 0, y la ecuación (3) tendrá la forma

$$y_{n+1} = y_n + h(\gamma_0 f_n + \gamma_1 \Delta f_{n-1})$$

substituyendo $\gamma_0 = 1$, $\gamma_1 = ½$ y teniendo $\Delta f_{n-1} = f_n - f_{n-1}$, tendremos

$$\begin{aligned} y_{n+1} &= y_n + hf_n + \frac{h}{2}(f_n - f_{n-1}) \\ &= y_n + hf_n + \frac{h}{2}f_n - \frac{h}{2}f_{n-1} \\ &= y_n - \frac{h}{2}f_{n-1} - \frac{3h}{2}f_n \\ &= y_n + \frac{h}{2}\left(-f(x_{n-1}, y_{n-1}) + 3f(x_n, y_n)\right) \end{aligned}$$

este algoritmo requiere conocer los puntos (x_n, y_n), y (x_1, y_1) para obtener el siguiente y así sucesivamente, ya que el primero de estos dos no esta dado, como se mencionó antes, el segundo punto puede calcularse con algún método de un paso.

MÉTODO DE ADAMS BASHFORTH DE TRES PASOS

Este método debe su nombre a que se utilizan tres puntos, esto se refleja al emplear la ecuación (3) del método de Adams-Bashforth de dos pasos considerando hasta el tercer término, tenemos

$$y_{n+1} = y_n + h(\gamma_0 f_n + \gamma_1 \Delta f_{n-1} + \gamma_2 \Delta^2 f_{n-2})$$

De manera similar al método anterior se substituyen γ_0, γ_1, γ_2 por sus valores correspondientes y además notando que

$$\Delta f_{n-1} = f_n - f_{n-1} \text{ y } \Delta^2 f_{n-2} = f_n - 2f_{n-1} + f_{n-2}$$

se llega a la siguiente ecuación.

$$y_{n+1} = y_n + h\left(f_n + \frac{1}{2}(f_n - f_{n-1}) + \frac{5}{12}(f_n - 2f_{n-1} + f_{n-2})\right)$$

desarrollando

$$y_{n+1} = y_n + hf_n + \frac{h}{2}f_n - \frac{h}{2}f_{n-1} + \frac{5h}{12}f_n - \frac{5h}{12}2f_{n-1} + \frac{5h}{12}f_{n-2}$$

$$y_{n+1} = y_n + \frac{23}{12}hf_n - \frac{16}{12}hf_{n-1} + \frac{5}{12}hf_{n-2}$$

$$y_{n+1} = y_n + \frac{h}{12}\left[23f(x_n, y_n) - 16f(x_{n-1}, y_{n-1}) + 5f(x_{n-2}, y_{n-2})\right]$$

El método necesita de tres puntos para iniciar la aproxomación, uno de conocido, (x_0,y_0) y los dos siguientes (x_1,y_1) y (x_2,y_2) calculados por alguno de los métodos de Euler o Runge-Kutta.

MÉTODO DE ADAMS-BASHFORTH DE CUATRO PASOS

Se obtiene como los anteriores considerando la ecuación (3) del método de Adams-Bashfoth de dos pasos pero empleando un término más que en el método anterior. De esta manera, nuestra ecuación toma la forma siguiente

$$y_{n+1} = y_n + \gamma_0 f_n + \gamma_1 \Delta f_{n-1} + \gamma_2 \Delta^2 f_{n-2} + \gamma_3 \Delta^3 f_{n-3}$$

Ahora substituimos los valores de las γ's y en lugar de las diferencias sus ordenadas correspondientes, obteniendo

$$y_{n+1} = y_n + h\left(f_n - \frac{1}{2}(f_n - f_{n-1}) - \frac{5}{12}(f_n - 2f_{n-1} + f_{n-2}) - \frac{3}{8}(f_n - 3f_{n-1} + 3f_{n-2} - f_{n-3})\right)$$

desarrollando

$$y_{n+1} = y_n + hf_n + \frac{h}{2}f_n - \frac{h}{2}f_{n-1} - \frac{5h}{12}f_n + \frac{10h}{12}f_{n-1} - \frac{5h}{12}f_{n-2} - \frac{3h}{8}f_n + \frac{9h}{8}f_{n-1} - \frac{9h}{8}f_{n-2} + \frac{3}{8}f_{n-3}$$

agrupando

$$y_{n+1} = y_n + \frac{55h}{24}f_n - \frac{59h}{24}f_{n-1} + \frac{37h}{24}f_{n-2} - \frac{9h}{24}f_{n-3}$$

esto es

$$y_{n+1} = y_n + \frac{h}{24}\left[55f(x_n, y_n) - 59f(x_{n-1}, y_{n-1}) + 37f(x_{n-2}, y_{n-2}) - 9f(x_{n-3}, y_{n-3})\right]$$

Como en los métodos previos la mecánica es la misma, dado un punto (x_0, y_0) calculamos los tres restantes, con algún método de un paso y con los cuatro puntos de apoyo resolvemos sucesivamente hasta obtener la aproximación a la solución.

MÉTODO DE ADAMS BASHFORTH DE CINCO PASOS

Nuevamente considerando la ecuación (3) del método de Adams-Bashforth de dos pasos pero ahora hasta su cuarto término, se tiene la siguiente expresión

$$y_{n+1} = y_n + h(\gamma_0 f_n + \gamma_1 \Delta f_{n-1} + \gamma_2 \Delta^2 f_{n-2} + \gamma_3 \Delta^3 f_{n-3} + \gamma_4 \Delta^4 f_{n-4})$$

De igual manera que para los métodos anteriores se substituyen tanto diferencias como coeficientes γ's por sus respectivos valores para obtener

$$y_{n+1} = y_n + h\left[f_n + \frac{1}{2}(f_n - f_{n-1}) + \frac{5}{12}(f_n - 2f_{n-1} + f_{n-2}) + \frac{3}{8}(f_n - 3f_{n-1} + 3f_{n-2} - f_{n-3}) + \frac{251}{720}(f_n - 4f_{n-1} + 6f_{n-2} - 4f_{n-3} + f_{n-4})\right]$$

desarrollando, y reagrupando, obtenemos

$$y_{n+1} = y_n + \frac{h}{720}\left[1901f(x_n, y_n) - 2774f(x_{n-1}, y_{n-1}) + 2616f(x_{n-2}, y_{n-2}) - 1274f(x_{n-3}, y_{n-3}) + 251f(x_{n-4}, y_{n-4})\right]$$

que es la ecuación de recursividad del método de Adams-Bashforth de cinco pasos. De manera similar a los métodos, se necesita un punto inicial conocido y cuatro calculados para posteriormente trabajar sucesivamente obteniendo la aproximación final a la solución.

Algoritmo: (Métodos de Adams-Bashforth 2, 3, 4 y 5 pasos)

Para aproximar la solución del problema de valor inicial

$$y' = f(t,y), \quad x_a \le t \le x_t, \quad y_a = y(x_a)$$

Entrada x_a, x_t puntos extremos, y_a condición inicial, n número de subintervalos y np número de pasos.

Salida $f(x_i,y_i)$ solución aproximada.

Paso 1 $h = (x_t - x_a)/n$

Paso 2 para $i = 0, \ldots, np-1$

$k1 = f(x_i,y_i)$

$k2 = f(x_i + h/2\ ,\ y_i + hk1/2)$

$k3 = f(x_i + h/2\ ,\ y_i + hk2/2)$

$k4 = f(x_i + h\ ,\ y_i + hk3)$

$y_{i+1} = y_i + h(k1 + 2k2 + 2k3 + k4)/6$

$x_{i+1} = x_i + h$

Paso 3 En caso de np, 2 Paso 4, 3 Paso 5, 4 Paso 6, 5 Paso 7

Paso 4 para $i = np-1, \ldots, n-1$

$G_i = (3f(x_i,y_i) - f(x_{i-1},y_{i-1}))/2$

Paso 5 para $i = np-1, \ldots, n-1$

$G_i = (23f(x_i,y_i) - 16f(x_{i-1},y_{i-1}) + 5f(x_{i-2},y_{i-2}))/12$

Paso 6 para $i = np-1, \ldots, n-1$

$G_i = (55f(x_i,y_i) + 59f(x_{i-1},y_{i-1}) + 37f(x_{i-2},y_{i-2})$
$- 9f(x_{i-3},y_{i-3}))/24$

Paso 7 para $i = np-1, \ldots, n-1$

$G_i = (1901f(x_i,y_i) - 2774f(x_{i-1},y_{i-1}) + 2616f(x_{i-2},y_{i-2}) - 1274f(x_{i-3},y_{i-3}) + 251f(x_{i-4},y_{i-4})$

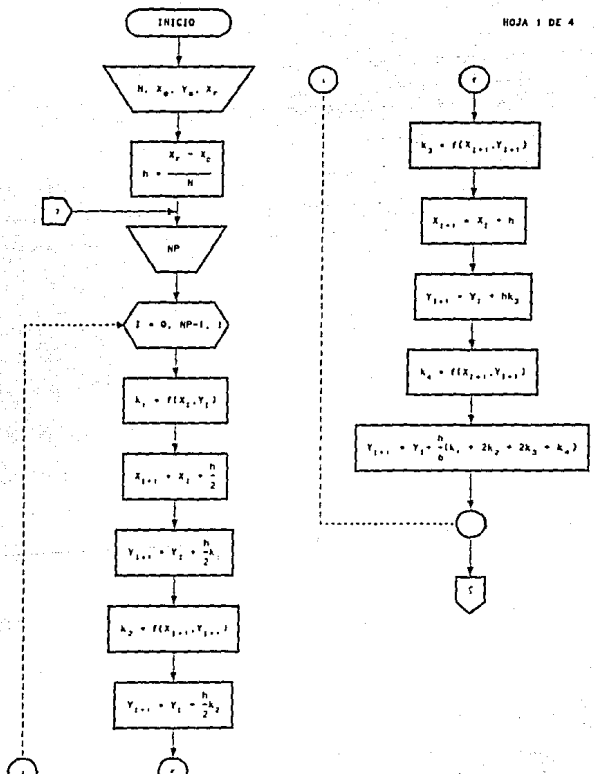
Paso 8 para $i = np-1, \ldots, n-1$

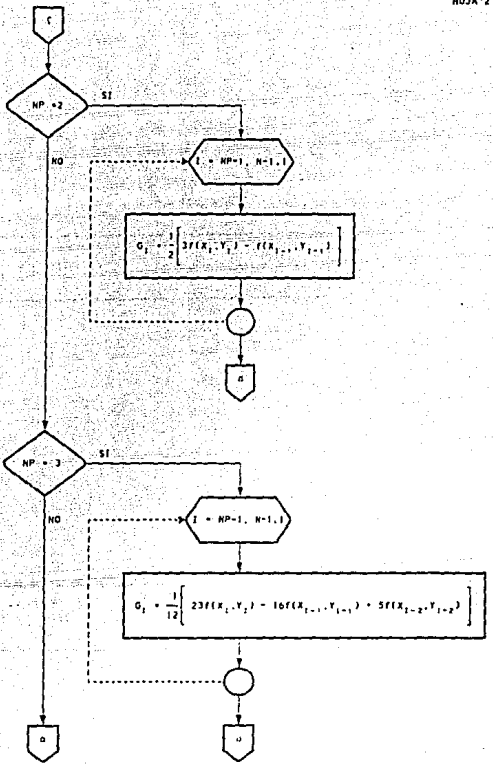
$y_{i+1} = y_i + G_i h$

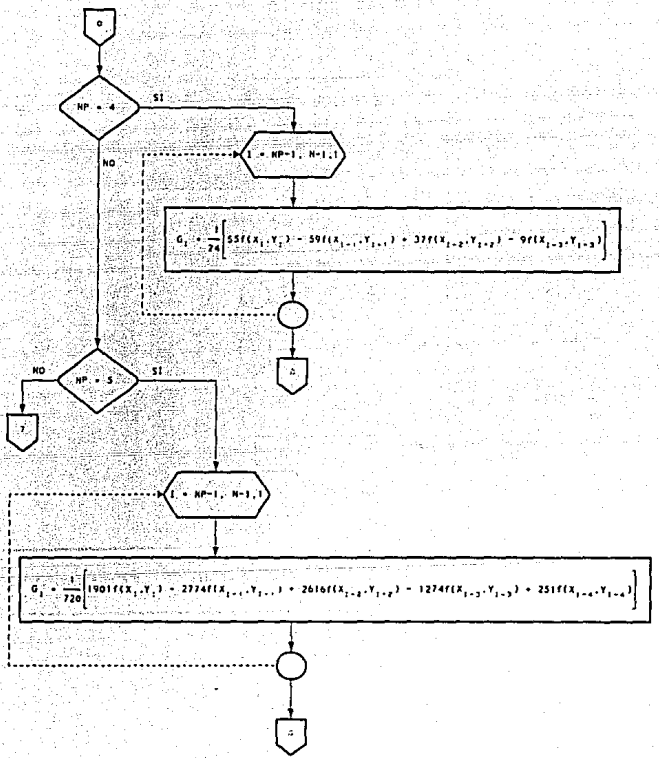
$x_{i+1} = x_i + h$

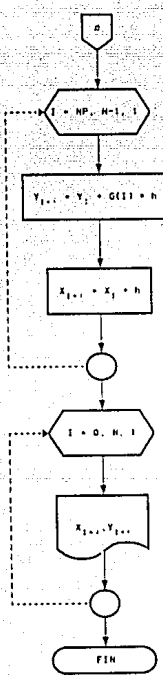
Paso 9 Salida (para $i = 0, \ldots, n-1$, $f(x_i,y_i)$) Terminar.

METODOS DE ADAMS BASHFORTH DE 2, 3, 4 Y 5 PASOS




C
NP =2
SI
NO
I = NP-1, N-1, 1
$G_I = \frac{1}{2}\left[3f(X_I,Y_I) - f(X_{I-1},Y_{I-1})\right]$
A
NP = 3
SI
NO
I = NP-1, N-1, 1
$G_I = \frac{1}{12}\left[23f(X_I,Y_I) - 16f(X_{I-1},Y_{I-1}) + 5f(X_{I-2},Y_{I-2})\right]$
B
B


O
HP = 4
SI
NO
I = HP-1, N-1,1
$G_i = \frac{1}{24}\left[55f(X_i,Y_i) - 59f(X_{i-1},Y_{i-1}) + 37f(X_{i-2},Y_{i-2}) - 9f(X_{i-3},Y_{i-3})\right]$
A
HP = 5
NO
SI
7
I = HP-1, N-1,1
$G_i = \frac{1}{720}\left[1901f(X_i,Y_i) - 2774f(X_{i-1},Y_{i-1}) + 2616f(X_{i-2},Y_{i-2}) - 1274f(X_{i-3},Y_{i-3}) + 251f(X_{i-4},Y_{i-4})\right]$
A


A
I = NP, N-1, 1
Y(I+1) = Y(I) + G(I) * h
X(I+1) = X(I) + h
I = 0, N, 1
X(I+1), Y(I+1)
FIN

```
Program Metodo_Adams_Bashfort;
uses Crt;
var X, Y  : array[0..15] of real;
    Xf, k1, k2, k3, k4, h : real;
    i, n  : integer;
    c : char;
    paso : byte;

function F(x, y : real) : real;
begin
  F := ((1+x)*sqr(y))/2;
end;


procedure Inicializa;
begin
  Xf := 0;
  k1 := 0;
  k2 := 0;
  k3 := 0;
  k4 := 0;
  h := 0;
  i := 0;
  n := 0;
  paso := 0;
  c := char(0);
end;


procedure Captura;
begin
  clrscr;
  gotoxy(30,3);write('METODO ADAMS-BASHFORT');
  gotoxy(17,5);write('Ecuación diferencial a evaluar: F(x,y) = x + 2xy');
  gotoxy(26,8);write('Suministre los siguientes datos');
  gotoxy(30,11);write('Valor inicial:  Xo = ');
  gotoxy(30,13);write('Valor inicial:  Yo = ');
  gotoxy(30,15);write('Valor final:    Xf = ');
  gotoxy(27,17);write('Número de intervalos: n = ');
  gotoxy(20,20);write('El número de pasos debe estar entre 2 y 5');
  gotoxy(29,22);write('Número de pasos:  p = ');
  gotoxy(20,24);write('Presione cualquier tecla para continuar');
  gotoxy(51,11);read(x[0]);
  gotoxy(51,13);read(y[0]);
  gotoxy(51,15);read(Xf);
  gotoxy(53,17);read(n);
end;
```

```
function G(paso, i: integer): real;
begin
 case paso of
   2: G := (-F(x[i-1],y[i-1]) + 3*F(x[i],y[i]))/2;
   3: G := (23*F(x[i],y[i]) - 16*F(x[i-1],y[i-1]) + 5*F(x[i-2],y[i-2]))/12;
   4: G := (55*F(x[i],y[i]) - 59*F(x[i-1],y[i-1]) + 37*F(x[i-2],y[i-2]) - 9*F(x[i-3],y[i-3]))/24;
   5: G := (1901*F(x[i],y[i])-2774*F(x[i-1],y[i-1])+ 2616*F(x[i-2],y[i-2])-1274*F(x[i-3],y[i-3])
          +251*F(x[i-4],y[i-4]))/720;
 end;
end;

procedure CalculaXY;
begin
 for i := 0 to paso-1 do
  begin
    k1 := F(x[i] , y[i]);
    x[i+1] := x[i]+h/2;
    y[i+1] := y[i]+(h*k1)/2;
    k2 := F(x[i+1] , y[i+1]);
    y[i+1] := y[i]+(h*k2)/2;
    k3 := F(x[i+1] , y[i+1]);
    x[i+1] := x[i]+h;
    y[i+1] := y[i]+h*k3;
    k4 := F(x[i+1] , y[i+1]);
    y[i+1] := y[i] + (h/6)*(k1 + 2*k2 + 2*k3 + k4);
  end;
 for i := paso to n-1 do
  begin
    y[i+1] := y[i] + G(paso,i)*h;
    x[i+1] := x[i] + h;
  end;
end;

procedure AB2345;
begin
 gotoxy(51,22);read(paso);
 if not((paso >= 2) and (paso <= 5)) then
   begin
     gotoxy(20,20);textcolor(0);textbackground(7);
     gotoxy(20,20);write('El número de pasos debe estar entre 2 y 5');
     gotoxy(20,20);textcolor(7);textbackground(0);
     AB2345;
   end;
 gotoxy(60,24);c := readkey;
 h := (Xf - x[0])/n;
 CalculaXY;
end;
```

```
procedure resultado;
begin
  clrscr;
  gotoxy(30,3);write('METODO ADAMS-BASHFORT');
  gotoxy(35,4);write('De ',paso,' pasos');
  gotoxy(17,5);write('Ecuación diferencial evaluada: F(x,y) = x + 2xy');
  gotoxy(13,7);write('en el intervalo ',x[0]:5:5,' < = x < = ',XF:5:5,
       ' con h = ',h:5:5);
  if n < = 7 then
   begin
    for i := 0 to n do
     begin
      gotoxy(23,9+i*2);write('Punto P',i,' = (',x[i]:5:8,',',y[i]:5:8,')');
     end;
    gotoxy(20,24);write('Presione cualquier tecla para continuar');
    gotoxy(60,24);c := readkey;
   end
  else
    begin
     for i := 0 to 7 do
     begin
      gotoxy(23,9+i*2);write('Punto P',i,' = (',x[i]:5:8,',',y[i]:5:8,')');
     end;
     gotoxy(20,24);write('Presione cualquier tecla para continuar');
     gotoxy(60,24);c := readkey;
     for i := 0 to 7 do
     begin
      gotoxy(20,9+i*2);clreol;
     end;
     for i := 8 to n do
      begin
       gotoxy(23,7+(i-7)*2);write('Punto P',i,' = (',x[i]:5:8,',',y[i]:5:8,')');
      end;
     gotoxy(20,24);write('Presione cualquier tecla para continuar');
     gotoxy(60,24);c := readkey;
    end;
end;

begin
  Inicializa;
  Captura;
  AB2345;
  Resultado;
end.
```

MÉTODOS MULTIPASO DE INTEGRACIÓN CERRADA

Estos métodos de integración también utilizan las diferencias finitas sin dividir hacia atrás pero con la salvedad de que se evalúan incluso a partir del punto (x_{n+1},y_{n+1}) indicando que la obtención de y_{n+1} esta en función del punto (x_{n+1}, y_{n+1}) que es desconocido, debido a esto se les llama métodos de integración cerrada.

NOTA: El algoritmo del método de Adams-Moulton se presenta junto con el del método de Milne de 4° y 6° grado, y de la misma manera el programa abarca a los tres métodos.

MÉTODO DE ADAMS-MOULTON DE DOS PASOS

Se obtienen utilizando diferencias finitas sin dividir hacia atrás interpolando $x_{n+1}, x_n, \ldots, x_{n-m}$ para un entero $m > 0$, las diferencias que interpolan en estos $m+2$ puntos en términos de $s=(x - x_n)/h$ es

$$P_{n+1}(s) = \sum_{k=0}^{m+1} (-1)^k \binom{1-s}{k} \Delta^k f_{n+1-k}$$

Estas diferencias están basadas en los valores de $f_{n+1}, f_n, \ldots, f_{n-m}$. Integrando en el intervalo $[x_n, x_{n+1}]$ y utilizamos

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} f(x, y(x))dx$$

obtenemos

$$y_{n+1} = y_n + h(\gamma^*_0 f_{n+1} + \gamma^*_1 \Delta f_n + \ldots + \gamma^*_{m+1} \Delta^{m+1} f_{n-m} \quad (1)$$

donde

$$\gamma^*_k = (-1)^k \int_0^1 \binom{1-s}{k} ds, \quad k = 0, 1, \ldots, m+1$$

los primeros valores de γ^*_k son

$$\gamma^*_0 = 1, \quad \gamma^*_1 = -\frac{1}{2}, \quad \gamma^*_2 = -\frac{1}{12}, \quad \gamma^*_3 = -\frac{1}{24}, \quad \gamma^*_4 = -\frac{19}{720}$$

para el método de dos pasos, emplearemos la ecuación (1), tomando en cuenta hasta el tercer

término, tenemos

$$y_{n+1} = y_n + h(\gamma^*_0 f_{n+1} + \gamma^*_1 \Delta f_n + \gamma^*_2 \Delta^2 f_{n-1})$$

substituimos los valores de las γ's y en lugar de trabajar con diferencias lo hacemos con ordenadas

$$y_{n+1} = y_n + h\left(f_{n+1} - \frac{1}{2}(f_{n+1} - f_n) - \frac{1}{12}(f_{n+1} - 2f_n + f_{n-1})\right)$$

desarrollando

$$y_{n+1} = y_n + hf_{n+1} - \frac{h}{2}f_{n+1} - \frac{h}{2}f_n - \frac{h}{12}f_{n+1} + \frac{2h}{12}f_n - \frac{h}{12}f_{n-1}$$

reagrupando

$$y_{n+1} = y_n + \frac{5}{12}hf_{n+1} - \frac{8}{12}hf_n + \frac{1}{12}hf_{n-1}$$

$$y_{n+1} = y_n + \frac{h}{12}(5f_{n+1} + 8f_n - f_{n-1})$$

$$y_{n+1} = y_n + \frac{h}{12}[5f(x_{n+1}, y_{n+1}) + 8f(x_n, y_n) - f(x_{n-1}, y_{n-1})]$$

Esta ecuación representa el método de Adams-Moulton de dos pasos.

Debido a que desconocemos y_{n+1} estos métodos no se utilizan directamente, si no que deberán ir acompañados de otro, el cual irá prediciendo los valores de y_{n+1}, para que este método vaya corrigiendo los valores de y_{n+1}.

MÉTODO DE ADAMS-MOULTON DE TRES PASOS

Como se realizó con el método de Adams-Moulton de dos pasos, empleamos la ecuación (1) de dicho método, tomando en cuenta hasta el cuarto término para obtener una ecuación de la forma.

$$y_{n-1} = y_n + h(\gamma'_0 f_{n-1} + \gamma'_1 f_n + \gamma'_2 \Delta^2 f_{n-1} + \gamma'_3 \Delta^3 f_{n-2})$$

ahora intercambiamos los valores por sus correspondientes coeficientes γ'_s y desarrollamos

$$y_{n-1} = y_n + h\left(f_{n-1} - \frac{1}{2}(f_{n-1} - f_n) - \frac{1}{12}(f_{n-1} - 2f_n + f_{n-1}) - \frac{1}{24}(f_{n-1} - 3f_n + 3f_{n-1} - f_{n-2})\right)$$

$$y_{n-1} = y_n + h\left(f_{n-1} - \frac{1}{2}f_{n-1} - \frac{1}{2}f_n - \frac{1}{12}f_{n-1} + \frac{2}{12}f_n - \frac{1}{12}f_{n-1} - \frac{1}{24}f_{n-1} + \frac{3}{24}f_n - \frac{3}{24}f_{n-1} + \frac{1}{24}f_{n-2}\right)$$

$$y_{n-1} = y_n + h\left(\frac{9}{24}f_{n-1} - \frac{19}{24}f_n - \frac{5}{24}f_{n-1} + \frac{1}{24}f_{n-2}\right)$$

cambiando a ordenadas y factorizando

$$y_{n-1} = y_n + \frac{h}{24}\left[9f(x_{n-1}, y_{n-1}) + 19f(x_n, y_n) - 5f(x_{n-1}, y_{n-1}) + f(x_{n-2}, y_{n-2})\right]$$

Esta última expresión representa a este método. Dicha expresión debe ir acompañada de otra ecuación que prediga valores ya que éste método se utiliza únicamente como corrector.

MÉTODO DE ADAMS-MOULTON DE CUATRO PASOS

Para obtener la ecuación que representa a este método, utilizamos la ecuación (1) del método de Adams-Moulton de dos pasos, consideramos un término adicional al método anterior

$$y_{n+1} = y_n + h(\gamma'_0 f_{n+1} + \gamma'_1 \Delta f_n + \gamma'_2 \Delta^2 f_{n-1} + \gamma'_3 \Delta^3 f_{n-2} + \gamma'_4 \Delta^4 f_{n-3})$$

De forma similar que para los otros métodos anteriores substituimos γ's, desarrollamos y reagrupamos:

$$y_{n+1} = y_n + \frac{h}{720}[251f_{n+1} + 646f_n - 264f_{n-1} + 106f_{n-2} - 19f_{n-3}]$$

ahora intercambiamos por ordenadas

$$y_{n+1} = y_n + \frac{h}{720}[251f(x_{n+1}, y_{n+1}) + 646f(x_n, y_n) - 264f(x_{n-1}, y_{n-1}) + 106f(x_{n-2}, y_{n-2}) - 19f(x_{n-3}, y_{n-3})]$$

para obtener el algoritmo de cuatro pasos para el método de Adams-Moulton. Esta es claramente una fórmula correctora de tipo cerrado ya que $f_{n+1} = f(x_{n+1}, y_{n+1})$ incluye la cantidad desconocida y_{n+1}. Debe por consiguiente resolverse por iteración.

MÉTODOS PREDICTORES CORRECTORES

Una característica de los métodos de Runge-Kutta consiste en que para obtener el siguiente punto, x_{m+1},y_{m+1}, usamos la información suministrada por x_m,y_m pero no por los puntos precedentes. En los métodos multipaso tenemos que evaluar la función en uno o más puntos adicionales. Esto no es tan razonable, pues después de varios pasos del proceso de integración disponemos de información adicional sin ninguna valuación de funciones, ya que se conoce el valor de y en los puntos previos. El hecho de que los métodos de un paso no utilizan esta información los margina un poco. Sin embargo los métodos siguientes no pueden iniciar por si solos (p. ej. métodos de integración cerrada). Ya que requieren utilizar algunos puntos previos. Por lo anterior, se puede usar algún método de paso sencillo para iniciar el proceso. Esto nos lleva a una combinación de métodos.

Estos métodos, de los que existen muchas variantes, se denominan métodos predictores-correctores. Esto indica que primero "predecimos" un valor de y_{m+1}. Después usamos una fórmula diferente para "corregir" este valor. Podremos utilizar esta última fórmula para "recorregir" el valor de y_{m+1}. Este proceso puede iterarse hasta cumplir con algún criterio y finalizar.

MÉTODO DE MILNE DE CUARTO GRADO Y SEXTO GRADO

El método de Milne es un método de pasos múltiples, que primero pronostica un valor para y_{n+1}, extrapolando los valores de la derivada. Difiere del de Adams en que corrige el valor predicho antes de pasar al siguiente paso. Los valores que se requieren para el inicio se pueden calcular mediante el método de Runge-Kutta, posiblemente por la serie de Taylor. En el método de Milne, se supone que para el inicio se conocen cuatro puntos equidistantes que son x_n, x_{n-1}, x_{n-2}, x_{n-3}. Se pueden emplear las fórmulas de cuadratura para integrar de la siguiente manera

$$\frac{dy}{dx} = f(x, y)$$

$$\int_{\cdots}^{\cdots} \left(\frac{dy}{dx}\right)_{dx} = \int_{\cdots}^{\cdots} f(x, y(x))dx = \int_{\cdots}^{\cdots} P(x)dx$$

de manera que el predictor será

$$y_{n+1} = y_{n-1} + \frac{4h}{3}(2f_n - f_{n-1} + 2f_{n-2})$$

Se integra la función f(x,y) reemplazándola con un polinomio de interpolación cuadrática que se ajuste a los tres puntos, en donde $x = x_n$, x_{n-1} y x_{n-2}. Note que se extrapola en la integración por un panel tanto a la izquierda como a la derecha de la región de ajuste. Con este valor de y_{n+1} se puede calcular f_{n+1} con una precisión razonable. La fórmula anterior es la

fórmula predictora y la correctora se obtiene como sigue

$$\int_{x_{n-1}}^{x_{n+1}} \left(\frac{dy}{dx}\right)_m = \int_{x_{n-1}}^{x_{n+1}} f(x,y)\,dx = \int_{x_{n-1}}^{x_{n+1}} P(x)\,dx$$

$$y_{n+1} = y_{n-1} + \frac{h}{3}(f_{n+1} + 4f_n + f_{n-1})$$

En la ecuación correctora el polinomio P no es idéntico al de la ecuación predictora, debido a que no ajusta en los mismos tres puntos, en la función correctora el polinomio se ajusta en x_{n+1}, x_n, x_{n-1}. Estos cambios son debidos a que no se extrapola el polinomio; f_{n+1} se calcula utilizando y_{n+1} para la fórmula predictora. La fórmula de integración es la ya conocida regla de Simpson de un tercio, ya que se integra un polinomio cuadrático sobre dos paneles dentro de la región de ajuste.

Y la fórmula para el método de Milne de 6° es la siguiente

$$y_{n+1} = y_{n-5} + \frac{3h}{10}[11f(x_n, y_n) - 14f(x_{n-1}, y_{n-1}) + 26f(x_{n-2}, y_{n-2}) - 14f(x_{n-3}, y_{n-3}) + 11f(x_{n-4}, y_{n-4})]$$

$$y_{n+1} = y_{n-3} + \frac{2h}{45}[7f(x_{n+1}, y_{n+1}) + 32f(x_n, y_n) + 12f(x_{n-1}, y_{n-1}) + 32f(x_{n-2}, y_{n-2}) + 7f(x_{n-3}, y_{n-3})]$$

MÉTODO DE ADAMS MOULTON O ADAMS MODIFICADO

Un método predictor conveniente para usarlo con el método de Adams-Moulton de cuatro pasos o cuarto orden es la fórmula de Adams-Bashforth de cuatro pasos. En este caso el predictor es del mismo orden que el corrector. Si se escoge un tamaño apropiado de paso, h, entonces la aplicación del corrector conducirá a una mejora significativa en la precisión. Para la ecuación diferencial $y' = f(x,y)$ con h fijo y $x_n\ x_{n-1} + nh$ y con (x_0,f_0), (x_1,f_1), (x_2,f_2), (x_3,f_3) dados, para cada $n = 3, 4, \ldots$fijo:

calcula los siguientes pasos

$$y^{*}_{n+1} = y_n + \frac{h}{24}\left(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}\right)$$

$$f^{*}_{n+1} = f\left(x_{n+1}, y_{n+1}\right)$$

$$y_{n+1} = y_n + \frac{h}{24}\left(9f\left(x_{n+1}, y^{(k-1)}_{n+1}\right) + 19f_n - 5f_{n-1} + f_{n-2}\right)$$

con $k = 1, 2, \ldots$

Este algoritmo produce una precisión mejorada, la fórmula correctora también nos puede dar un estimado del error, que puede usarse para decidir si el paso h es adecuado para la precisión requerida, tomando como criterio de convergencia el error relativo.

Un método que no tiene el mismo problema de inestabilidad, pero que tiene la misma eficiencia, es el método de Adams-Moulton o Adams modificado. Este también supone un conjunto de valores iniciales ya calculados por alguna otra técnica. Aquí se considera un

polinomio cúbico a través de cuatro puntos, de x_{n-3} hasta x_n y se integra sobre un sólo paso de x_n hasta x_{n+1}.

$$\int_{x_n}^{x_{n+1}} \left(\frac{dy}{dx}\right)_{x_n} = \int_{x_n}^{x_{n+1}} f(x, y)\,dx = \int_{x_n}^{x_{n+1}} P(x)\,dx$$

$$y_{n+1} = y_n + h\left(f_n - \frac{1}{2}\Delta f_{n-1} + \frac{5}{12}\Delta^2 f_{n-2} + \frac{3}{8}\Delta^3 f_{n-3}\right)$$

$$y_{n+1} = y_n + \frac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})$$

La fórmula de integración se obtiene escribiendo a P(x) como un polinomio interpolante de retroceso de Newton-Gregory (ajustando en x_n, x_{n-1}, x_{n-2}, x_{n-3}) e integrando. Utilizando la ecuación anterior como "predictor", se calcula el valor correcto de f_{n+1}. Si ahora se aproxima a f(x,y) como un polinomio cúbico que se ajusta en el intervalo de x_{n-2} hasta x_{n+1} y se integra de x_n hasta x_{n+1}, no se estará extrapolando al polinomio, teniendo un término con menor error. El resultado es

$$y_{n+1} = y_n + h\left(f_{n+1} - \frac{1}{2}\Delta f_n - \frac{1}{12}\Delta^2 f_{n-1} - \frac{1}{24}\Delta^3 f_{n-2}\right)$$

$$y_{n+1} = y_n + \frac{h}{24}(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2})$$

Algoritmo: **(Métodos Predictores Correctores Adams-Moulton, Milne de 4º y 6º)**

Para aproximar la solución del problema de valor inicial

$$y' = f(t,y), \quad x_o \le t \le x_t, \quad y_o = y(x_o)$$

Entrada x_o, x_t puntos extremos, y_o condición inicial, n número de subintervalos y m método a emplear.

Salida $f(x_i,y_i)$ solución aproximada.

Paso 1 $h = (x_t - x_o)/n$

Paso 2 En caso OP igual a 1 hacer 2.1, 2 hacer 2.2, 3 hacer 2.3

2.1: OP = 3, IP = 3, IC = 1

2.2: OP = 5, IP = 5, IC = 3

2.3: OP = 4, IP = 0, IC = 0

Paso 2 para i = 0, ..., OP-1

$k1 = f(x_i,y_i)$

$k2 = f(x_i + h/2\ ,\ y_i + hk1/2)$

$k3 = f(x_i + h/2\ ,\ y_i + hk2/2)$

$k4 = f(x_i + h\ ,\ y_i + hk3)$

$y_{i+1} = y_i + h(k1 + 2k2 + 2k3 + k4)/6$

$x_{i+1} = x_i + h$

Paso 3 En caso de m, 1 Paso 4, 2 Paso 5, 3 Paso 6

Paso 4 para i = OP ..., n-1

$x_{i+1} = x_i + h$

$y_{i+1} = y_{i-IP} + MP(OP,I)*h$

$y_{i+1} = y_{i-IC} + MC(OP,I)*h$

Paso 5 Salida(para i = 0, ..., n, f(x,y))

Terminar.

Continua en la siguiente página

Se tomará una función MP(OP,I) para las ecuaciones predictoras y para las ecuaciones correctoras otra función llamada MC(OP,I).

FUNCION MP(OP,I)

Paso A En caso de OP igual a 3 paso B, 5 paso C, 4 paso D

B: $MP = 4(2f(x_i,y_i) - f(x_{i-1},y_{i-1}) + 2f(x_{i-2},y_{i-2}))/3$

C: $MP = 3(11f(x_i,y_i) - 14f(x_{i-1},y_{i-1}) + 26f(x_{i-2},y_{i-2})$
$- 14f(x_{i-3}, y_{i-3}) + 11f(x_{i-4},y_{i-4}))/10$

D: $MP = (55f(x_i,y_i) - 59f(x_{i-1},y_{i-1}) + 37f(x_{i-2},y_{i-2})$
$- 9f(x_{i-3},y_{i-3}))/24$

Paso E Regresa.

FUNCION MC(OP,I)

Paso A En caso de OP igual a 3 paso B, 5 paso C, 4 paso D

B: $MC = (f(x_{i-1},y_{i-1}) + 4f(x_i,y_i) + f(x_{i+1},y_{i+1}))/3$

C: $MC = 2(7f(x_{i+1},y_{i+1}) + 32f(x_i,y_i) + 12f(x_{i-1},y_{i-1})$
$+ 32f(x_{i-2},y_{i-2}) + 7f(x_{i-3},y_{i-3}))/45$

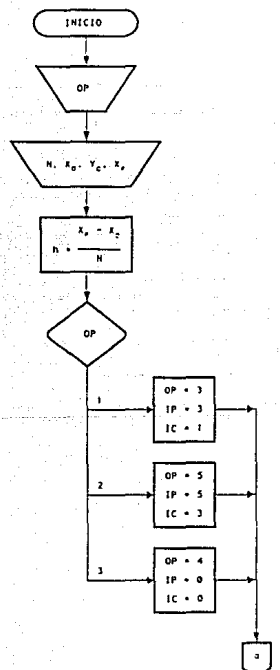
D: $MC = (9f(x_{i+1},y_{i+1}) + 19f(x_i,y_i) - 5f(x_{i-1},y_{i-1})$
$+ f(x_{i-2}, y_{i-2}))/24$

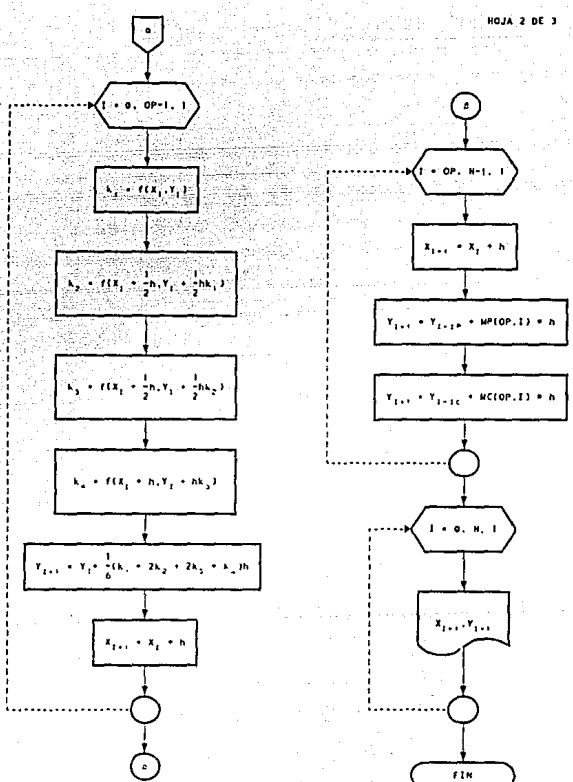
Paso E Regresa.

METODOS PREDICTORES CORRECTORES: ADAMS-MOULTON

MILNE 4° GRADO

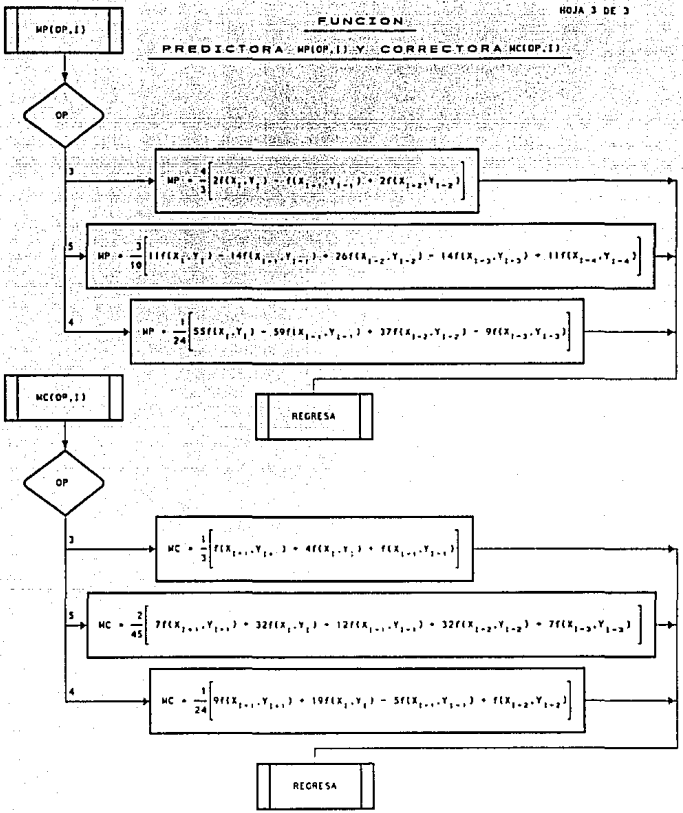
MILNE 6° GRADO




a
I = 0, OP-1, 1
$k_1 = f(X_I, Y_I)$
$k_2 = f(X_I + \frac{1}{2}h, Y_I + \frac{1}{2}hk_1)$
$k_3 = f(X_I + \frac{1}{2}h, Y_I + \frac{1}{2}hk_2)$
$k_4 = f(X_I + h, Y_I + hk_3)$
$Y_{I+1} = Y_I + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h$
$X_{I+1} = X_I + h$
c
b
I = OP, N-1, 1
$X_{I+1} = X_I + h$
$Y_{I+1} = Y_{I-1P} + MP(OP,I) * h$
$Y_{I+1} = Y_{I-1C} + MC(OP,I) * h$
I = 0, N, 1
X_{I+1}, Y_{I+1}
FIN

FUNCION

PREDICTORA MP(OP,I) Y CORRECTORA MC(OP,I)



```
Program Metodos_Predictores_Correctores;
uses Crt;
var X, Y  : array[0..15] of real;
    Xf, k1, k2, k3, k4, h : real;
    i, m, n, Ip , Ic  : integer;
    c  : char;
    Op : byte;

function F(x, y : real) : real;
begin
  F := ((1 + x)*sqr(y))/2;
end;

procedure Inicializa;
begin
  Xf := 0;
  k1 := 0;
  k2 := 0;
  k3 := 0;
  k4 := 0;
  h  := 0;
  i  := 0;
  n  := 0;
  Op := 0;
  c  := char(0);
end;

procedure Datos;
begin
  clrscr;
  gotoxy(25,6);write('METODOS PREDICTORES CORRECTORES');
  gotoxy(17,8);
  write('Ecuación diferencial a evaluar: F(x,y) = x + 2xy');
  gotoxy(20,11);write('Suministre los  siguientes datos');
  gotoxy(30,14);write('Valor inicial:  Xo = ');
  gotoxy(30,16);write('Valor inicial:  Yo = ');
  gotoxy(30,18);write('Valor final:    Xf = ');
  gotoxy(27,20);write('Número de intervalos: n = ');
  gotoxy(20,24);write('Presione  cualquier tecla para continuar');
  gotoxy(51,14);read(x[0]);
  gotoxy(51,16);read(y[0]);
  gotoxy(51,18);read(Xf);
  gotoxy(53,20);read(n);
  gotoxy(20,24);write('Presione  cualquier tecla para continuar');
  gotoxy(60,24);c := readkey;
end;

procedure Revisa;
begin
  gotoxy(43,22);read(Op);
```

```
  if not((Op >= 1) and (Op <= 3)) then
    begin
      gotoxy(20,20);textcolor(0);textbackground(7);
      gotoxy(19,20);write('La opción del método debe estar entre 1 y 3');
      gotoxy(20,20);textcolor(7);textbackground(0);
      Revisa;
    end
  else
    begin
      gotoxy(60,24);c := readkey;
      Datos;
    end;
end;

procedure Captura;
begin
  clrscr;
  gotoxy(25,3);write('METODOS PREDICTORES CORRECTORES');
  gotoxy(17,5);write('Ecuación diferencial a evaluar: F(x,y) = x + 2xy');
  gotoxy(26,8);write('Suministre los siguientes datos');
  gotoxy(30,12);write('1.- Milne de 4° orden');
  gotoxy(30,14);write('2.- Milne de 6° orden');
  gotoxy(30,16);write('3.- Adams - Moulton');
  gotoxy(19,20);write('La opción del método debe estar entre 1 y 3');
  gotoxy(35,22);write('Opción [_]');
  gotoxy(20,24);write('Presione cualquier tecla para continuar');
  Revisa;
end;

function MP(Op, i: integer): real;
var nm :real;
begin
  case Op of
    3: MP := (4/3)*(2*F(x[i],y[i]) - F(x[i-1],y[i-1]) + 2*F(x[i-2],y[i-2]));
    5: MP := (3/10)*(11*F(x[i],y[i])-14*F(x[i-1],y[i-1]) + 26*F(x[i-2],y[i-2])-14*F(x[i-3],y[i-3])
          + 11*F(x[i-4],y[i-4]));
    4: MP := (55*F(x[i],y[i])-59*F(x[i-1],y[i-1])+37f(x[-2],y[i-2])-9*F(x[i-3],y[i-3]))/24;
  end;
end;

function MC(Op, i: integer): real;
begin
  case Op of
    3: MC := (F(x[i+1],y[i+1]) + 4*F(x[i],y[i]) + F(x[i-1],y[i-1]))/3;
    5: MC := (2/45)*(7*F(x[i+1],y[i+1])+32*F(x[i],y[i]) + 12*F(x[i-1],y[i-1])+32*F(x[i-2],y[i-2])
          + 7*F(x[i-3],y[i-3]));
    4: MC := (9*F(x[i+1],y[i+1]) + 19*F(x[i],y[i]) - 5*F(x[i-1],y[i-1]) + F(x[i-2],y[i-2]))/24;
  end;
end;
```

```
procedure CalculaXY;
begin
  for i := 0 to Op-1 do
   begin
     k1 := F(x[i] , y[i]);
     k2 := F(x[i]+h/2 , y[i]+(h*k1)/2);
     k3 := F(x[i]+h/2 , y[i]+(h*k2)/2);
     k4 := F(x[i]+h , y[i]+h*k3);
     y[i+1] := y[i] + (h/6)*(k1 + 2*k2 + 2*k3 + k4);
     x[i+1] := x[i] + h;
   end;
  for i := Op to n-1 do
   begin
     x[i+1] := x[i] + h;
     y[i+1] := y[i-ip] + MP(Op,i)*h;
     y[i+1] := y[i-ic] + MC(Op,i)*h;
   end;
end;

procedure MPC;
begin
  h := (Xf - x[0])/n;
  case Op of
    1: begin
         Op := 3;
         Ip := 3;
         Ic := 1;
         CalculaXY;
         clrscr;gotoxy(25,3);write('METODOS PREDICTORES CORRECTORES');
         gotoxy(32,4);write('Milne de 4° orden');
       end;
    2: begin
         Op := 5;
         Ip := 5;
         Ic := 3;
         CalculaXY;
         clrscr;gotoxy(25,3);write('METODOS PREDICTORES CORRECTORES');
         gotoxy(32,4);write('Milne de 6° orden');
       end;
    3: begin
         Op := 4;
         Ip := 0;
         Ic := 0;
         CalculaXY;
         clrscr;gotoxy(25,3);write('METODOS PREDICTORES CORRECTORES');
         gotoxy(32,4);write('Adams - Moulton');
       end;
  end;
end;
```

```
procedure resultado;
begin
 gotoxy(17,5);write('Ecuación diferencial evaluada: F(x,y) = x + 2xy');
 gotoxy(13,7);write('en el intervalo '+x[0]:5:5+' <= x <= '+X[n]:5:5+' con h = '+h:5:5);
 if n <= 7 then
  begin
   for i := 0 to n do
    begin
     gotoxy(23,9+i*2);write('Punto P'+i+' = ('+x[i]:5:8+' , '+y[i]:5:8+')');
    end;
   gotoxy(20,24);write('Presione cualquier tecla para continuar');
   gotoxy(60,24);c := readkey;
  end
  else
   begin
    for i := 0 to 7 do
    begin
     gotoxy(23,9+i*2);
     write('Punto P'+i+' = ('+x[i]:5:8+' , '+y[i]:5:8+')');
    end;
    gotoxy(20,24);write('Presione cualquier tecla para continuar');
    gotoxy(60,24);c := readkey;
    for i := 0 to 7 do
    begin
     gotoxy(20,9+i*2);clreol;
    end;
    for i := 8 to n do
     begin
      gotoxy(23,7+(i-7)*2);write('Punto P'+i+' = ('+x[i]:5:8+' , '+y[i]:5:8+')');
     end;
    gotoxy(20,24);write('Presione cualquier tecla para continuar');
    gotoxy(60,24);c := readkey;
   end;
end;

begin
 Inicializa;
 Captura;
 MPC;
 Resultado;
end.
```

ECUACIONES DIFERENCIALES: PROBLEMAS DE VALOR DE FRONTERA

Los métodos de la pasada sección se usan para resolver ecuaciones diferenciales de primer orden, los cuales deben satisfacer una condición inicial al principio de la integración. Consideraremos ahora ecuaciones diferenciales de orden superior a uno, en las que se establezcan dos condiciones en los extremos del intervalo de integración.

A este tipo de problemas se les conoce como problemas de valores de frontera. Su solución numérica es más difícil que la de los problemas con condiciones iniciales, ya que la solución debe satisfacer a ambas condiciones.

MÉTODO DE DIFERENCIAS FINITAS LINEALES

El presente resultado se utiliza para resolver ecuaciones diferenciales de cualquier orden con condiciones iniciales o con condiciones de frontera. Consiste en dividir el intervalo de solución en n subintervalos, a cada uno de los puntos que definen cada subintervalo se les llama punto pivote. A continuación se substituyen las derivadas de la ecuación diferencial por fórmulas de derivación numérica, todas ellas con el mismo orden de error.

La ecuación de diferencias finitas que resulta, se puede aplicar en forma recursiva para cada uno de los puntos pivote que forman el intervalo solución, resolviéndose en términos de la solución previamente obtenida con el mismo procedimiento en los puntos pivote anteriores.

El error que se comete al utilizar este método depende del error en las fórmulas de derivación que se utilicen. Para una ecuación diferencial de cualquier orden pero lineal, el procedimiento anterior se puede modificar de tal modo que se obtenga un sistema de ecuaciones algebraicas simultáneas en lugar de suponer valores de las ordenadas.

Entonces la solución de un problema ordinario lineal de valores de frontera por diferencias finitas reduce la integración de la ecuación diferencial a la evaluación de las raíces de un sistema de ecuaciones algebraicas simultáneas. Estas raíces son los valores de la solución requerida en los puntos pivote de su intervalo de definición.

SISTEMAS DE ECUACIONES DIFERENCIALES ORDINARIAS

Las ecuaciones diferenciales ordinarias involucran una función desconocida. Consecuentemente, en cada paso hemos resuelto una ecuación diferencial la cual involucra una función desconocida. Por muchas razones, incluyendo aplicaciones y generalizaciones, uno está interesado en el estudio de n ecuaciones diferenciales con n funciones desconocidas donde n es un entero mayor o igual a dos.

Tales sistemas aparecen de manera natural en problemas que involucran varias variables dependientes, cada una de las cuales es una función de una sola variable independiente. Muchos sistemas físicos o biológicos importantes involucran, como se menciono antes, variables interrelacionadas; por ejemplo: las magnitudes de corrientes y voltajes en diferentes ramales de un circuito eléctrico, la posición de las mallas en una red fuente, y la concentración de solutos en varias células de un organismo. En algunos casos, uno puede aproximar las leyes que gobiernan el comportamiento de estos sistemas mediante un modelo diferencial lineal. Esto es, un modelo el cual asume que la rapidez del crecimiento de cada, x'_i, de cada variable, x_i, es una combinación lineal de las otras variables.

$$x'_i = a_{i1}x_1 + \ldots + a_{in}x_n$$

donde a_{ij}, $i,j = 1, \ldots, n$ son dos operadores diferenciales lineales definidos en un intervalo; x_i,

..., x_n son funciones desconocidas de la variable independiente y n es el número de variables en el sistema modelado por las ecuaciones.

MÉTODO DE EULER

Los métodos de Taylor, Euler, Runge-Kutta, etc., también pueden ser aplicados a problemas de valor inicial para un sistema de ecuaciones diferenciales de primer orden. Para esto consideraremos un sistema con dos funciones desconocidas, excepto en los métodos aplicados a sistemas con cualquier número de funciones desconocidas.

De esta manera, consideraremos el problema de valor inicial

$$x' = f_1(t,x,y)$$

$$y' = f_2(t,x,y) \qquad (1)$$

$$x(t_o) = x_o, \qquad y(t_o) = y_o \qquad (2)$$

En este sistema x y y son las funciones desconocidas y t es la variable independiente. Para una solución numérica del problema de valor inicial (1) y (2) en el intervalo $t_o <= t <= b$, pensando en un conjunto de puntos $\{(t_i, x_i, y_i)\}^n$ $i = 0$ donde $t_n = b$ y x_i, y_i son aproximaciones a la solución del sistema (1) cuando $t = t_i$. Asumiendo que Δt_i es la misma para todas las i; de modo que, $\Delta t_i = = (b - t_o)/n$.

Para el problema de valor inicial (1) - (2). El método de Euler queda como sigue:

$$x_i = x_{i-1} + f_1(t_{i-1}, x_{i-1}, y_{i-1})h, \quad i = 1, 2, \ldots, n$$

$$y_i = y_{i-1} + f_2(t_{i-1}, x_{i-1}, y_{i-1})h, \quad i = 1, 2, \ldots, n$$

MÉTODO DE TAYLOR

Ahora consideraremos el método de Taylor correspondiente a sistemas. Las fórmulas del método son bastante complicadas en general.

$$\frac{\partial^2 y}{\partial t^2}_{i,j} = \frac{d}{dt}\left[y'(t)\right]_{i,j} = \frac{d}{dt}\left[f_2(t, x, y)\right]_{i,j}$$

$$= \left[\frac{\partial f_2}{\partial t} + \frac{\partial f_2}{\partial x}\frac{dx}{dt} + \frac{\partial f_2}{\partial y}\frac{dy}{dt}\right]_{i,j}$$

$$= \frac{\partial f_2}{\partial t}(t_n, x_n, y_n) + \frac{\partial f_2}{\partial x}(t_n, x_n, y_n) \cdot f_1(t_n, x_n, y_n) + \frac{\partial f_2}{\partial y}(t_n, x_n, y_n) \cdot f_2(t_n, x_n, y_n)$$

Naturalmente, el alto orden de la aproximación de Taylor, se refleja en una mayor complejidad en los cálculos. A continuación se presentan las fórmulas para una expansión en la serie de Taylor de orden dos.

$$x_{i+1} = x_i + f_1(t_i, x_i, y_i)h + \left[\frac{\partial f_1}{\partial t}(t_i, x_i, y_i) + \frac{\partial f_1}{\partial x}(t_i, x_i, y_i) \cdot f_1(t_i, x_i, y_i) + \frac{\partial f_1}{\partial y}(t_i, x_i, y_i) \cdot (t_i, x_i, y_i)\right]\frac{h^2}{2}$$

$$y_{i+1} = y_i + f_2(t_i, x_i, y_i)h + \left[\frac{\partial f_2}{\partial t}(t_i, x_i, y_i) + \frac{\partial f_2}{\partial x}(t_i, x_i, y_i) \cdot f_1(t_i, x_i, y_i) + \frac{\partial f_2}{\partial y}(t_i, x_i, y_i) \cdot (t_i, x_i, y_i)\right]\frac{h^2}{2}$$

MÉTODO DE RUNGE - KUTTA

Para el método de Runge-Kutta, la generalización para sistemas es más sencilla. Por ejemplo, la aproximación de tercer orden debe ser como sigue:

$$
\begin{aligned}
\alpha_1 &= hf_1(t_i, x_i, y_i) \\
\beta_1 &= hf_2(t_i, x_i, y_i) \\
\alpha_2 &= hf_1\left(t_i + \frac{1}{2}h, x_i + \frac{1}{2}\alpha_1, y_i + \beta_1\right) \\
\beta_2 &= hf_2\left(t_i + \frac{1}{2}h, x_i + \frac{1}{2}\alpha_1, y_i + \beta_1\right) \\
\alpha_3 &= hf_1(t_i + h, x_i + 2\alpha_2 - \alpha_1, y_i + 2\beta_2 - \beta_1) \\
\beta_3 &= hf_2(t_i + h, x_i + 2\alpha_2 - \alpha_1, y_i + 2\beta_2 - \beta_1) \\
x_{i+1} &= x_i + \frac{1}{6}(\alpha_1 + 4\alpha_2 + \alpha_3) \\
y_{i+1} &= y_i + \frac{1}{6}(\beta_1 + 4\beta_2 + \beta_3)
\end{aligned}
$$

La generalización para sistemas con el método de Runge-Kutta de cuarto orden se obtiene de manera similar.

CAPITULO VI

SISTEMAS DE ECUACIONES ALGEBRAICAS

SISTEMAS DE ECUACIONES ALGEBRAICAS LINEALES

En el área de Ingeniería así como en algunas otras, frecuentemente encontramos problemas que involucran la resolución de sistemas de ecuaciones lineales, ya sea porque provienen directamente de la formulación del problema; en muchos casos es una estrategia o parte del ataque de otro tipo de problema. Estos problemas se presentan en áreas como transferencia de calor, transferencia de masa, análisis de circuitos eléctricos, etc. El objetivo es solucionar sistemas de n ecuaciones lineales simultáneas con n incógnitas

$$\sum_{j=1}^{n} a_{ij} x_j, \quad i = 1, 2, \ldots, n$$

La ecuación anterior escrita en forma matricial es la siguiente

$$Ax = b$$

donde $A = [a_{ij}]$ es la matriz de coeficientes $x^T = (x_1, \ldots, x_n)$ y $b^T(b_1, \ldots, b_n)$ con la T denotando la traspuesta.

Las matrices que se presentan generalmente en la práctica quedan en una de las siguientes categorías:

- Cubiertas pero no grandes. Cubierta denota tener pocos elementos que valgan cero y no grandes denota matrices de orden menor a 30. Tales matrices se presentan en una amplia variedad de problemas en Estadística, Física e Ingeniería, etc.

- Dispersas y quizá muy grandes. En contraste con las anteriores estas matrices tienen pocos elementos diferentes de cero. En muchos casos, estos elementos quedan cerca o sobre la diagonal principal. Muy grandes significa un orden de 100 o mayor. Tales matrices se originan normalmente en la solución numérica de ecuaciones diferenciales parciales.

Básicamente son los diferentes caracteres (cubiertas y dispersas) de las matrices lo que hace que los métodos directos sean generalmente superiores para la primera categoría, mientras que los métodos iterativos sean usados con mayor frecuencia para problemas de la segunda categoría. Sin embargo, se enfatiza que no existen reglas precisas en este sentido. Es poco recomiendable el uso de métodos iterativos para matrices cubiertas de bajo orden, no obstante, se utilizan métodos directos, aún para matrices dispersas grandes.

MÉTODOS ITERATIVOS O SUBSTITUCIÓN DIRECTA

Muchos problemas en la práctica requieren de la solución de grandes sistemas de ecuaciones lineales, teniendo en mayor proporción coeficientes cero en esos sistemas. Esta clase de sistemas se presentan con frecuencia en la solución de problemas que implican ecuaciones diferenciales parciales o en la solución de problemas de valores en la frontera, y el objetivo es obtener tales soluciones, esto motiva la investigación de estos métodos.

Un método iterativo para resolver un sistema de ecuaciones lineales $Ax = b$ de orden n empieza con una aproximación inicial $x^{(0)}$ a la solución x, y genera una sucesión de vectores que converge a x. La mayoría de estos métodos convierten el sistema $Ax = b$ en uno equivalente de la forma $x = Tx + c$. Habiendo elegido el vector inicial $x^{(0)}$. La sucesión de vectores de soluciones aproximadas se genera calculando

$$x^{(k)} = Tx^{(k-1)} + c, \quad k = 1, 2, \ldots$$

La idea básica que encierran los métodos iterativos es, en esencia, la misma que la de la iteración de punto fijo. Estas técnicas se emplean rara vez para resolver sistemas pequeños pues presentan gran desventaja respecto a los métodos directos en lo que se refiere al tiempo requerido para obtener una aproximación con la suficiente precisión, no así para sistemas grandes como se mencionó anteriormente.

MÉTODO DE JACOBI

Se tiene un sistema de n ecuaciones algebraicas con n incógnitas

$$\begin{aligned} a_{11} x_1 + a_{12} x_2 + a_{13} x_3 + \ldots + a_{1n} x_n &= b_1 \\ a_{21} x_1 + a_{22} x_2 + a_{23} x_3 + \ldots + a_{2n} x_n &= b_2 \\ \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \\ a_{n1} x_1 + a_{n2} x_2 + a_{n3} x_3 + \ldots + a_{nn} x_n &= b_n \end{aligned}$$

Las fórmulas de iteración o recursión para el método de Jacobi implican un reordenamiento mediante el cual estas fórmulas puedan calcular un nuevo valor para cada ecuación. La manera más fácil es despejar por ejemplo x_1 de la primera ecuación obteniendo

$$x_1 = \frac{b_1 - a_{12} x_2 - a_{13} x_3 - \ldots - a_{1n} x_n}{a_{11}}$$

que conduce a la ecuación de recursión

$$x_1^{(k+1)} = \frac{b_1 - a_{12} x_2^{(k)} - a_{13} x_3^{(k)} - \ldots - a_{1n} x_n^{(k)}}{a_{11}}$$

en general, para x_i puede obtenerse la ecuación de iteración

$$x_i^{(k+1)} = \frac{b_i - a_{i1} x_1^{(k)} - \ldots - a_{i,i-1} x_{i-1}^{(k)} - a_{i,i+1} x_{i+1}^{(k)} \ldots - a_{in} x_n^{(k)}}{a_{ii}}$$

esta ecuación puede generar una secuencia de aproximaciones que converja a la solución exacta del sistema de ecuaciones lineales, iniciando con una aproximación arbitraria (x_1^0, x_2^0, x_3^0, ...

$x_n^{(0)}$). Si no se dispone de información sobre la solución, podremos tomar

$$x_i^{(0)} = \frac{b_i}{a_{ii}} \qquad i = 1, 2, 3, \ldots, n$$

Se determina la siguiente iteración $x^{(k+1)}$ a partir de $x^{(k)}$, $k = 0, 1, 2, \ldots$, mediante la ecuación de iteración para $i = 1, 2, \ldots, n$. Si el módulo de cada una de las diferencias

$$x_i^{(k+1)} - x_i^{(k)}, \qquad i = 1, 2, \ldots, n$$

es menor que la tolerancia establecida, entonces se detiene la iteración y se toma la última $x^{(k+1)}$ como la solución. En otro caso volvemos a determinar la siguiente iteración, substituyendo k por k+1.

Algoritmo

Para resolver Ax = b con una aproximación inicial dada $x^{(0)}$.

Entrada: Orden del sistema n;

Los elementos $a_{i,j}$ de la matriz A;

Las componentes b_i del término no homogéneo b;

Los elementos x^0_i de la aproximación inicial $x^{(0)}$,

$a_{i,j}$, b_i, x_{0i} dentro de $1 \leq i \leq n$.

Salida: La solución aproximada $x_1, \ldots, x_n$.

Paso 1 Para i = 1, ..., n realizar

Paso 2 Tomar $xc_i = 0$

Paso 3 Para j = 1, ... n realizar

si j = i entonces paso 3

$xc_i = xc_i + a_{n,i} * x_j$

Paso 4 tomar

$$xc_i = \frac{a_{j,n+1} - xc_j}{a_{j,j}}$$

Paso 5 Si $| xc_i - x_i | < 1e\text{-}5$ entonces in = 0

Paso 6 Si in < > 0 entonces

para i = 1, ... n hacer

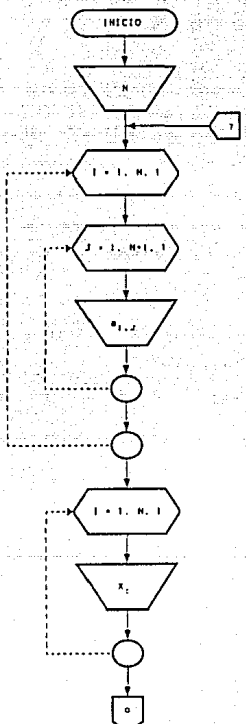
$x_i = xc_i$

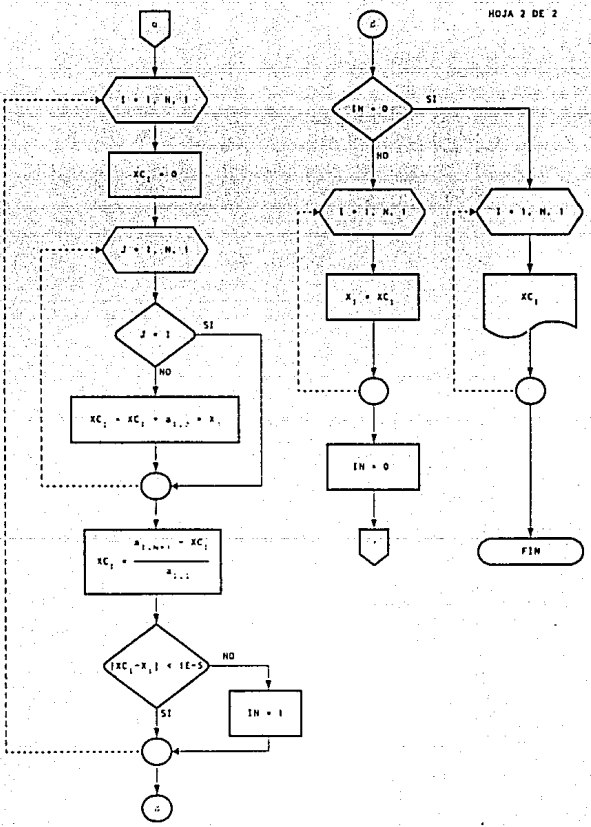
regresar a Entrada

Paso 7 Salida (para i = 1, ... n: Imprimir xc_i)

Terminar.

METODO DE JACOBI




a
I = 1, N, 1
XC_I = 0
J = 1, N, 1
J = I
SI
NO
XC_I = XC_I + a_I,J * X_J
XC_I = (a_I,N+1 - XC_I) / a_I,I
|XC_I - X_I| < 1E-5
NO
SI
IN = 1
a
IN = 0
SI
NO
I = 1, N, 1
X_I = XC_I
IN = 0
a
I = 1, N, 1
XC_I
FIN

```
Program Metodo_de_Jacobi;
uses Crt;
const P = 1e-10;
var a : array[1..30,1..30] of real;
  x, d, xc : array[1..30] of real;
  v, i, j, n, y, en : integer;
  z : real;
  c : char;

procedure Inicializa;
begin
 v := 0;
 i := 0;
 en:= 0;
 j := 0;
 n := 0;
 z := 0;
 y := 0;
 c := char(0);
end;

procedure Captura;
begin
 clrscr;
 gotoxy(32,3);write('METODO DE  JACOBI');
 gotoxy(18,5);write('Resolución de sistemas de ecuaciones lineales');
 gotoxy(29,8);write('Orden del sistema: n = ');
 gotoxy(52,8);read(n);
 z := (80 - n*10)/2 - 2*(n-1);
 y := trunc(z);
 for i := 1 to n do
  for j := 1 to n+1 do
   begin
    gotoxy(y+12*(j-1)-3,10+i*2);write('A[',i,',',j,'] = ');
   end;
 for i := 1 to n do
  for j := 1 to n+1 do
   begin
    gotoxy(y+12*(j-1)+6,10+i*2);read(a[i,j]);
   end;
 gotoxy(20,10+n*2+3);write('Suministre valores supuestos para cada x:');
 for i := 1 to n do
  begin
   gotoxy(y+12*(i-1)+6,10+n*2+5);write('X',i,' = ');
  end;
 for i := 1 to n do
  begin
   gotoxy(y+12*(i-1)+11,10+n*2+5);read(x[i]);
  end;
 d := x;
```

```
  gotoxy(20,24);write('Presione  cualquier tecla para continuar');
  gotoxy(60,24);c := readkey;
end;

procedure Jacobi;
begin
  for i := 1 to n do
    begin
      xc[i] := 0;
      for j := 1 to n do
        begin
          if j <> i then xc[i] := xc[i] + a[i,j]*x[j];
        end;
      xc[i] := (a[i,n+1] - xc[i]) / a[i,i];
      if not(ABS(xc[i] - x[i]) < P) then en := 1
                                    else en := 0;
    end;
  if en = 1 then
              begin
                v:=v + 1;
                x := xc;
                en := 0;
                Jacobi;
              end;
end;

procedure Resultado;
begin
  clrscr;
  gotoxy(32,3);write('METODO DE  JACOBI');
  gotoxy(18,5);write('Resolución de sistemas de ecuaciones lineales');
  gotoxy(29,7);write('Orden del sistema: n = ',n);
  z := (80 - n*10)/2 - 2*(n-1);
  y := trunc(z);
  for i := 1 to n do
    for j := 1 to n+1 do
      begin
        gotoxy(y+12*(j-1)-3.7+i*2);write('A[',i,',',j,'] = ');
      end;
  for i := 1 to n do
    for j := 1 to n+1 do
      begin
        gotoxy(y +12*(j-1)+6.7+i*2);write(a[i,j]:2:0);
      end;
  gotoxy(26.7+n*2+3);write('Valores supuestos para cada x');
  for i := 1 to n do
    begin
      gotoxy((y-n)+12*(i-1)+6.7+n*2+5);write('X',i,' = ');
    end;
```

```
  for i := 1 to n do
    begin
      gotoxy((y-n)+12*(i-1)+11,7+n*2+5);write(d[i,1]:2:2);
    end;
  gotoxy(27,7+n*2+7);write('La solución del sistema es:');
  for i := 1 to n do
    begin
      gotoxy((y-n)+17*(i-1),7+n*2+9);write('X',i,' = ',xc[i]:4:6);
    end;
  gotoxy(20,24);write('Presione  cualquier tecla para continuar');
  gotoxy(60,24);c := readkey;
end;

begin
  Inicializa;
  Captura;
  Jacobi;
  Resultado;
end.
```

MÉTODO DE GAUSS - SEIDEL

Este método iterativo sirve para resolver sistemas de ecuaciones lineales algebraicas, es una modificación del método de Jacobi. Al igual que otros métodos, éste aprovecha toda la información disponible ya que al emplear la ecuación de iteración

$$x_i^{(k+1)} = \frac{b_i - a_{i1}\, x_1^{(k)} - \ldots - a_{i,i-1}\, x_{i-1}^{(k)} - a_{i,i+1}\, x_{i+1}^{(k)} - \ldots - a_{in}\, x_n^{(k)}}{a_{ii}}$$

para calcular una nueva $x_i^{(k+1)}$, ya se han calculado los valores de $x_1^{(k+1)}$, $x_2^{(k+1)}$, ..., $x_{i-1}^{(k+1)}$. Por tanto, para emplear esta información, se puede modificar la ecuación de iteración, obteniendo la nueva ecuación de iteración

$$x_i^{(k+1)} = \frac{b_i - a_{i1}\, x_1^{(k+1)} - \ldots - a_{i,i-1}\, x_{i-1}^{(k+1)} - a_{i,i+1}\, x_{i+1}^{(k)} - \ldots - a_{in}\, x_n^{(k)}}{a_{ii}}$$

para i igual a 1, 2, 3, ..., n.

Esta ecuación se conoce como iteración de Gauss-Seidel. Al tomar el valor de uno el contador i, el lado derecho sólo tendrá términos con exponente (k) y cuando i = n, sólo términos con exponente (k+1). Durante el proceso, las operaciones sobre la matriz de coeficientes se realizan por renglones y para cada uno de ellos involucramos un elemento de b, requiriéndose varios elementos de x, siendo éste el caso tanto para los métodos iterativos como para la eliminación gausiana.

En la mayoría de los casos, el método de Gauss-Seidel converge con mayor rapidez que el método de Jacobi.

Algoritmo

Para resolver $Ax = b$ con una aproximación inicial dada $x^{(0)}$.

Entrada: Orden del sistema n;

Los elementos $a_{i,j}$ de la matriz A;

Las componentes b_i del término no homogéneo b;

Los elementos x^0_i de la aproximación inicial $x^{(0)}$;

$a_{i,j}$, b_i, x^0_i dentro de $1 \leq i \leq n$.

Salida: La solución aproximada $x_1, \ldots, x_n$.

Paso 1 Para $i = 1, \ldots, n$ realizar

Paso 2 Tomar suma = 0

Paso 3 Para $j = 1, \ldots, n$ realizar

Si $j = i$ entonces paso 3

Tomar suma = suma + $a_{i,j} * x_j$

Paso 4 Tomar

$$xc_i = \frac{a_{i,n+1} - suma}{a_{i,i}}$$

Paso 5 Si $|$ suma - x_i $|$ < 1e-5 entonces in = 0

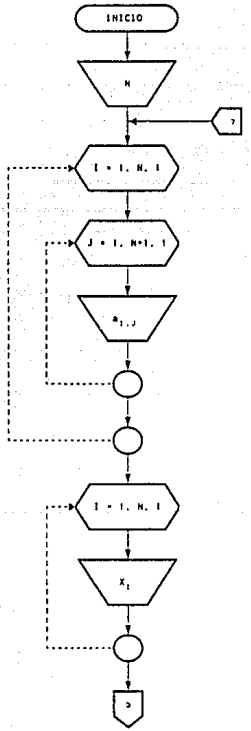
Paso 6 Si in < > 0 entonces n = 0

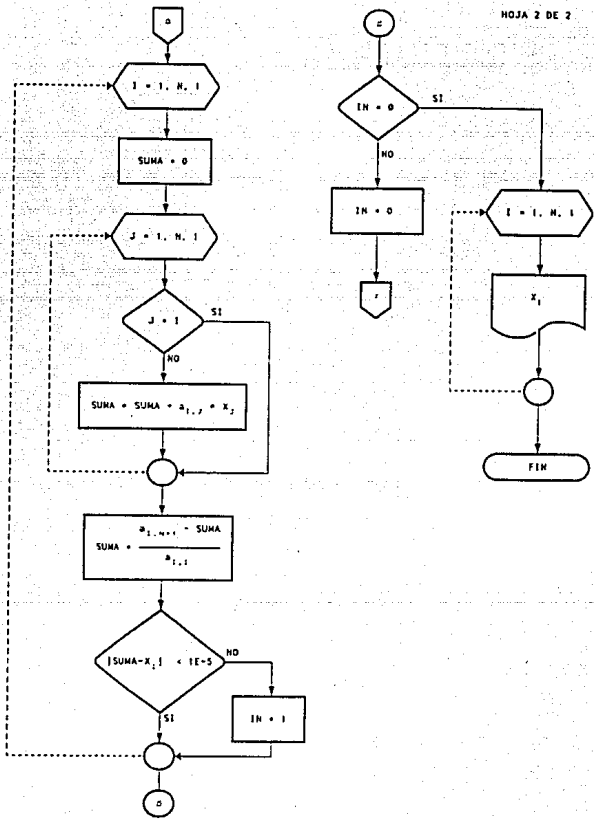
regresar a Entrada

Paso 7 Salida (para $i = 1, \ldots, n$: Imprimir xc_i)

Terminar.

METODO DE GAUSS-SEIDEL




a
I = 1, N, 1
SUMA = 0
J = 1, N, 1
J = I
SI
NO
SUMA = SUMA + $a_{I,J} * X_J$
$SUMA = \frac{a_{I,N+1} - SUMA}{a_{I,I}}$
|SUMA-X_I| < 1E-5
NO
SI
IN = 1
a
c
IN = 0
SI
NO
IN = 0
I = 1, N, 1
X_I
r
FIN

```
Program Metodo_de_Gauss_Seidel;
uses Crt;
const P = 1e-10;
var a : array[1..30,1..30] of real;
   x, d : array[1..30] of real;
   v, i, j, n, y, en : integer;
   z, suma : real;
   c : char;

procedure Inicializa;
begin
  v := 0;
  i := 0;
  en:= 0;
  j := 0;
  n := 0;
  z := 0;
  suma := 0;
  y := 0;
  c := char(0);
end;

procedure Captura;
begin
 clrscr;
 gotoxy(29,3);writeln('METODO DE  GAUSS-SEIDEL');
 gotoxy(18,5);writeln('Resolución de sistemas de ecuaciones lineales');
 gotoxy(29,8);write('Orden del sistema: n = ');
 gotoxy(52,8);read(n);
 z := (80 - n*10)/2 - 2*(n-1);
 y := trunc(z);
 for i := 1 to n do
  for j := 1 to n + 1 do
    begin
      gotoxy(y + 12*(j-1)-3,10+i*2);write('A[',i,',',j,'] = ');
    end;
 for i := 1 to n do
  for j := 1 to n + 1 do
    begin
      gotoxy(y + 12*(j-1) + 6,10 + i*2);readln(a[i,j]);
    end;
 gotoxy(20,10+n*2+3);writeln('Suministre valores supuestos para cada x:');
 for i := 1 to n do
   begin
     gotoxy(y + 12*(i-1) - 6,10 + n*2 - 5);write('X',i,' = ');
   end;
 for i := 1 to n do
   begin
     gotoxy(y + 12*(i-1) + 11,10 + n*2 + 5);read(x[i]);
   end;
 d := x;
 gotoxy(20,24);write('Presione  cualquier tecla para continuar');
 gotoxy(60,24);c := readkey;
end;


procedure GaussSeidel;
begin
 for i := 1 to n do
```

```
  begin
    suma := 0;
    for j := 1 to n do
      if j <> i then suma := suma + a[i,j]*x[j];
    suma := (a[i,n+1] - suma) / a[i,i];
    if not(ABS(suma - x[i]) < P) then en := 1;
    x[i] := suma;
  end;
 if en = 1 then
            begin
              v := v + 1;
              en := 0;
              GaussSeidel;
            end;
end;

procedure Resultado;
begin
 clrscr;
 gotoxy(29,3);write('METODO DE  GAUSS-SEIDEL');
 gotoxy(18,5);write('Resolución de sistemas de ecuaciones lineales');
 gotoxy(29,7);write('Orden del sistema: n = ',n);
 z := (80 - n*10)/2 - 2*(n-1);
 y := trunc(z);
 for i := 1 to n do
  for j := 1 to n+1 do
    begin
      gotoxy(y + 12*(j-1)-3,7+i*2);write('A[',i,',',j,'] = ');
    end;
 for i := 1 to n do
  for j := 1 to n-1 do
    begin
      gotoxy(y + 12*(j-1)+6,7+i*2);write(a[i,j]:2:0);
    end;
 gotoxy(26,7+n*2+3);write('valores supuestos para cada x');
 for i := 1 to n do
  begin
    gotoxy((y-n)+12*(i-1)+6,7+n*2+5);write('X',i,' = ');
  end;
 for i := 1 to n do
  begin
    gotoxy((y-n)+12*(i-1)+11,7+n*2+5);write(d[i]:2:2);
  end;
 gotoxy(27,7+n*2+7);write('La solución del sistema es:');
 for i := 1 to n do
  begin
    gotoxy((y-n)+17*(i-1),7+n*2+9);write('X',i,' = ',x[i]:4:6);
  end;
 gotoxy(20,24);write('Presione  cualquier tecla para continuar');
 gotoxy(60,24);c := readkey;
end;

begin
 Inicializa;
 Captura;
 GaussSeidel;
 Resultado;
end.
```

MÉTODOS DIRECTOS

Los métodos directos involucran alguna variación al procedimiento de la eliminación gaussiana. Estos métodos, al realizarse el proceso sin redondear resultados de cualquier operación, nos conducirán a la solución exacta del sistema dado.

ELIMINACIÓN GAUSSIANA

Se tiene el sistema

$$Ax = b$$

El sistema dado es transformado en pasos mediante rearreglos y combinaciones lineales apropiadas de las ecuaciones dentro del sistema siguiente

$$\begin{array}{l} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n = b_2 \\ \cdot \quad\quad \cdot \quad\quad \cdot \quad\quad\quad\quad \cdot \\ \cdot \quad\quad \cdot \quad\quad \cdot \quad\quad\quad\quad \cdot \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{nn}x_n = b_n \end{array}$$

formamos la matriz aumentada

$$\begin{array}{c} R_1: \\ R_2: \\ \cdot \\ \cdot \\ R_n: \end{array} \begin{bmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} & a_{1n+1} \\ a_{21} & a_{22} & a_{23} & \ldots & a_{2n} & a_{2n+1} \\ \cdot & \cdot & \cdot & \ldots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \ldots & \cdot & \cdot \\ a_{n1} & a_{n2} & a_{n3} & \ldots & a_{nn} & a_{nn+1} \end{bmatrix}$$

formada por los coeficientes de la matriz y además los elementos de b que son los elementos $a_{i,n+1}$ para cada i = 1, 2, ... n. Distinguiremos el renglón con el subíndice i y con la j a la columna.

Siempre que $a_{ii} <> 0$, se realiza lo siguiente:

- Dividir R_i por a_{ii} para que el nuevo elemento a_{ii} valga uno.
- Ahora multiplicamos el nuevo R_i por $a_{i+1,i}$ y lo restamos a R_{i+1}.

- Nuevamente multiplicamos R_i por a_{i+2}, y lo restamos a R_{i+2}, esto se repite hasta R_n.
- Para eliminar los coeficientes de x, en cada uno de estos renglones.

La matriz resultante será

$$\begin{array}{c} R_1: \\ R_2: \\ \cdot \\ \cdot \\ R_n: \end{array} \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & a_{1n+1} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n} & a_{2n+1} \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ 0 & 0 & 0 & 0 & a_{nn} & a_{nn+1} \end{bmatrix}$$

Esta matriz representa un sistema lineal con el mismo conjunto de soluciones que el sistema original. La matriz obtenida así, es una matriz triangular, la cual se puede resolver mediante la substitución hacia atrás. Resolviendo para R_n

$$x_n = \frac{a_{n\,n+1}}{a_{n\,n}}$$

y para las demás R_i

$$x_i = \frac{a_{i\,n+1} - \sum_{j=i+1}^{n} a_{ij}\, x_j}{a_{ii}}$$

para i igual n-1, n-2, n-3, ..., 1.

Algoritmo

Para resolver el sistema lineal de n x n

Entrada: Orden del sistema n;

Los elementos a_{ij} de la matriz aumentada A donde $1 < i < n$ y $1 < j < n+1$;

Los elementos $x''_1, \ldots, x''_n$ de la aproximación inicial

Salida: La solución aproximada $x_1, \ldots, x_n$.

Paso 1 Para $k = 1, \ldots, n-1$ realizar 2, 3, 4, 5, 6

Paso 2 Para $j = k, \ldots, n$ realizar

Si $a_{j,k} = 0$ entonces Salida(Matriz singular),

Terminar.

Paso 3 Si $j <> k$ entonces

Para $i = 1, \ldots, n+1$ realizar

$AA = A_{j,i}$

$A_{j,i} = A_{k,i}$

$A_{k,i} = AA$

Paso 4 Para $j = n+1, \ldots, k$ decrementado en 1 hacer

$A_{k,j} = A_{k,j} / A_{k,k}$

Paso 5 Para $i = k+1, \ldots, n$ realizar

Para $j = n+1, \ldots, 1$ decrementado en 1 hacer

$A_{i,j} = A_{i,j} - A_{k,j} * A_{i,k}$

Paso 7 Tomar $X_n = A_{n,n+1} / A_{n,n}$

Paso 8 Para $k = n-1, \ldots, 1$ decrementado en 1 realizar

Para $j = k+1, \ldots, n$ hacer

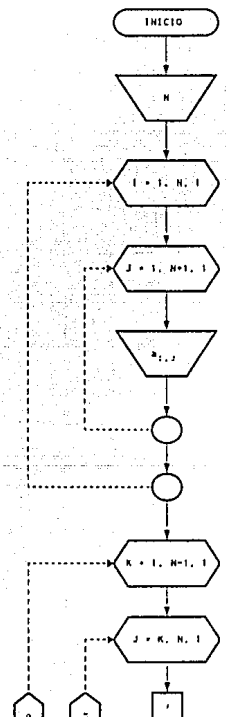
Tomar $x_k = x_k + A_{k,j} * x_j$

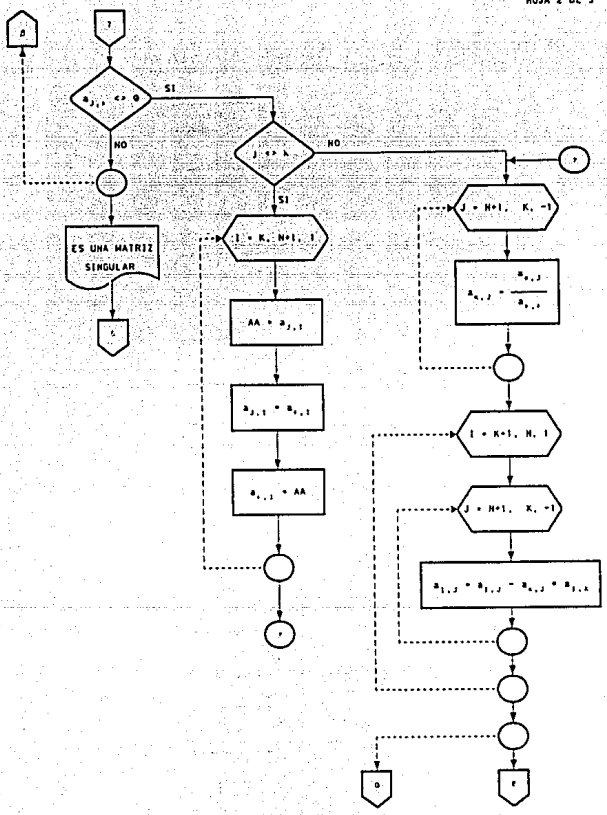
Tomar $x_k = A_{k,n+1} - x_k$

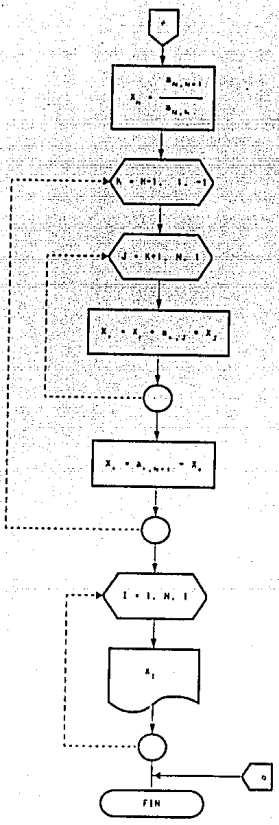
Paso 9 Salida (para $i = 1, \ldots, n$; Imprimir x_i)

Terminar.

METODO DE LA ELIMINACION GAUSSIANA




B
J
$a_{J,J}$ <> 0
SI
NO
J <> K
NO
SI
ES UNA MATRIZ SINGULAR
A
I = K, N+1, 1
AA = $a_{J,I}$
$a_{J,I} = a_{K,I}$
$a_{K,I} = AA$
C
J = N+1, K, -1
$a_{K,J} = \frac{a_{K,J}}{a_{K,K}}$
I = K+1, N, 1
J = N+1, K, -1
$a_{I,J} = a_{I,J} - a_{K,J} * a_{I,K}$
D
E


FIN

```
Program Metodo_de_Eliminacion_Gaussiana;
uses Crt;

var a,b : array[1..30,1..30] of real;
    x : array[1..30] of real;
    mensaje, i, j, k, n, y, en : integer;
    AA, z : real;
    c : char;

procedure Inicializa;
begin
  mensaje := 0;
  i := 0;
  j := 0;
  k := 0;
  n := 0;
  y := 0;
  en := 0;
  AA := 0;
  z := 0;
  c := char(0);
  for i := 1 to 30 do
   begin
    x[i] := 0;
   end;
end;

procedure Captura;
begin
  clrscr;
  gotoxy(30,3);write('ELIMINACION GAUSSIANA');
  gotoxy(18,5);write('Resolución de sistemas de ecuaciones lineales');
  gotoxy(29,8);write('Orden del sistema: n = ');
  gotoxy(52,8);read(n);
  z := (80 - n*10)/2 - 2*(n-1);
  y := trunc(z);
  for i := 1 to n do
   for j := 1 to n+1 do
     begin
       gotoxy(y + 12*(j-1)-3,10+i*2);write('A[',i,',',j,']= ');
     end;
  for i := 1 to n do
   for j := 1 to n+1 do
     begin
       gotoxy(y + 12*(j-1)+6,10+i*2);read(a[i,j]);
     end;
  b := a;
  gotoxy(20,24);write('Presione cualquier tecla para continuar');
  gotoxy(60,24);c := readkey;
end;
```

```
procedure Proceso;
begin
    for j := n+1 downto k do
      begin
        a[k,j] := a[k,j] / a[k,k];
      end;
    for i := k+1 to n do
      for j := n+1 downto k do
        begin
          a[i,j] := a[i,j] - a[k,j]*a[i,k];
        end;
end;

procedure Intercambio;
begin
    for j := k to n do
      if a[j,k] <> 0 then
                  begin
                    if j <> k then
                                begin
                                  for t := k to n+1 do
                                    begin
                                      AA := a[j,t];
                                      a[j,t] := a[k,t];
                                      a[k,t] := AA;
                                    end;
                                  proceso;
                                end
                               else
                                proceso;
                  end;
end;

procedure ElimGaussiana;
begin
  for k := 1 to n-1 do
   begin
     Intercambio;
   end;
  x[n] := a[n,n+1] / a[n,n];
  for k := n-1 downto 1 do
    begin
      for j := k+1 to n do
        begin
          x[k] := x[k] + a[k,j]*x[j];
        end;
      x[k] := a[k,n+1] - x[k];
    end;
end;
```

```
procedure Resultado;
begin
  clrscr;
  gotoxy(30,3);write('ELIMINACION GAUSSIANA');
  gotoxy(18,5);write('Resolución de sistemas de ecuaciones lineales');
  gotoxy(29,7);write('Orden del sistema: n = ',n);
  z := (80 - n*10)/2 - 2*(n-1);
  y := trunc(z);
  for i := 1 to n do
    for j := 1 to n+1 do
      begin
        gotoxy(y+12*(j-1)+3,7+i*2);write('A[',i,',',j,'] = ');
      end;
  for i := 1 to n do
    for j := 1 to n+1 do
      begin
        gotoxy(y+12*(j-1)+6,7+i*2);write(b[i,j]:2:0);
      end;
  if mensaje = 1 then
    begin
      gotoxy(25,7+n*2+3);write('Se trata de una matriz singular');
    end
  else
    begin
      gotoxy(27,7+n*2+7);write('La solución del sistema es:');
      for i := 1 to n do
        begin
          gotoxy((y-n)+17*(i-1),7+n*2+9);
          write('X',i,' = ',x[i]:4:6);
        end;
    end;
  gotoxy(20,24);write('Presione  cualquier tecla para continuar');
  gotoxy(60,24);c := readkey;
end;


begin
  Inicializa;
  Captura;
  ElimGaussiana;
  Resultado;
end.
```

MÉTODO DE GAUSS - JORDAN o DIAGONALIZACION

Este método constituye una variación del método de la eliminación gaussiana. El procedimiento trabaja sobre todas las ecuaciones restantes en el momento de eliminar la incógnita, o sea, tanto las anteriores como las posteriores a la ecuación pivote. En el método anterior R_i se empleó para la eliminar x_i de todas las ecuaciones bajo ella y sucesivamente con las demás R_i. En el presente método R_i también se utilizará para eliminar x_i de R_i. En el paso siguiente, R_i será la ecuación pivote, eliminando x_i tanto de R_i y R_i como de las situadas bajo ella. Eliminando así el empleo de la substitución hacia atrás, y obteniendo la matriz diagonal del sistema de ecuaciones lineales con la solución explícitamente dada.

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\
\cdot \quad\quad \cdot \quad\quad \cdot \quad\quad \cdot \quad\quad \cdot \quad\quad \cdot \quad\quad \cdot \quad\quad \cdot \quad\quad \cdot \\
\cdot \quad\quad \cdot \quad\quad \cdot \quad\quad \cdot \quad\quad \cdot \quad\quad \cdot \quad\quad \cdot \quad\quad \cdot \quad\quad \cdot \\
a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n &= b_n
\end{aligned}
$$

formando la matriz aumentada

$$
\left[\begin{array}{cccc|c}
a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & a_{1n+1} \\
a_{21} & a_{22} & a_{23} & \cdots & a_{2n} & a_{2n+1} \\
\cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\
a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} & a_{nn+1}
\end{array}\right]
$$

el procedimiento consta de una sola fase progresiva que se aplica a la misma matriz ampliada, teniéndose tantas transformaciones como ecuaciones. En cada etapa se transforma a una

columna de la matriz hasta finalmente obtener la matriz identidad

1a.

$$\left[\begin{array}{ccccc|c} 1 & a_{12} & a_{13} & \cdots & a_{1n} & a_{1n+1} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n} & a_{2n+1} \\ 0 & a_{32} & a_{33} & \cdots & a_{3n} & a_{3n+1} \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ 0 & a_{n2} & a_{n3} & \cdots & a_{nn} & a_{nn+1} \end{array}\right]$$

2n.

$$\left[\begin{array}{ccccc|c} 1 & 0 & a_{13} & \cdots & a_{1n} & a_{1n+1} \\ 0 & 1 & a_{23} & \cdots & a_{2n} & a_{2n+1} \\ 0 & 0 & a_{33} & \cdots & a_{3n} & a_{3n+1} \\ \cdot & \cdot & & \cdots & \cdot & \cdot \\ 0 & 0 & a_{n3} & \cdots & a_{nn} & a_{nn+1} \end{array}\right]$$

y así sucesivamente hasta n

n.

$$\left[\begin{array}{ccccc|c} 1 & 0 & 0 & \cdots & 0 & a_{1n+1} \\ 0 & 1 & 0 & \cdots & 0 & a_{2n+1} \\ 0 & 0 & 1 & \cdots & 0 & a_{3n+1} \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & 1 & a_{nn+1} \end{array}\right]$$

como se observa, la solución es evidente

Algoritmo:

Para resolver el sistema lineal de n x n

Entrada: Orden del sistema n;

Los elementos a_{ij} de la matriz aumentada A donde $1 < i < n$ y $1 < j < n+1$;

Salida: La solución aproximada $x_1, \ldots, x_n$.

Paso 1 Para k = 1, ..., n realizar 2, 3, 4, 5, 6

Paso 2 Para j = k, ..., n realizar

Si $a_{kk} = 0$ entonces Salida(Matriz singular),

Terminar.

Paso 3 Si j < > k entonces

Para i = k, ..., n+1 realizar

$AA = A_{ij}$

$A_{ij} = A_{kj}$

$A_{kj} = AA$

Paso 4 Para j = n+1, ..., k decrementado en 1 hacer

$A_{kj} = A_{kj} / A_{kk}$

Paso 5 Para i = 1, ..., n realizar

Si i < > k entonces

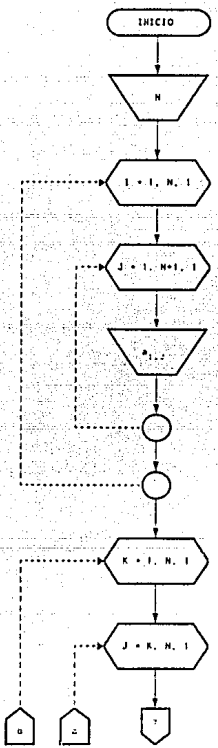
Para j = n+1, ..., k decrementado en 1 hacer

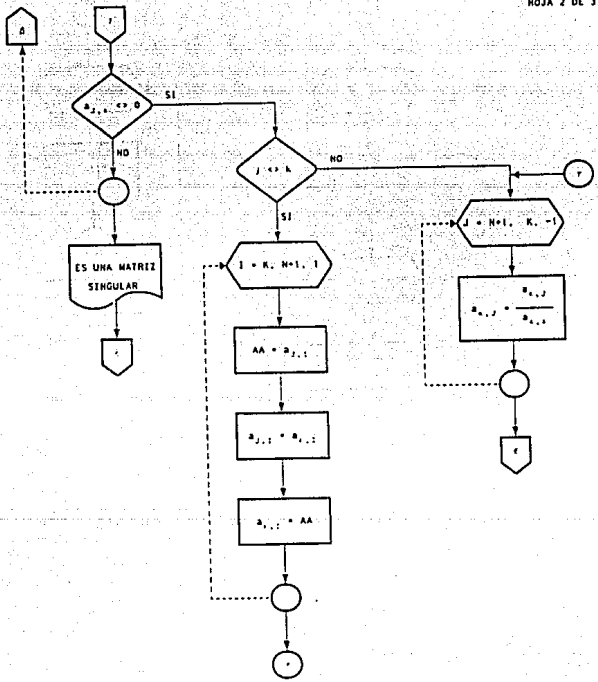
$A_{ij} = A_{ij} - A_{kj} * A_{ik}$

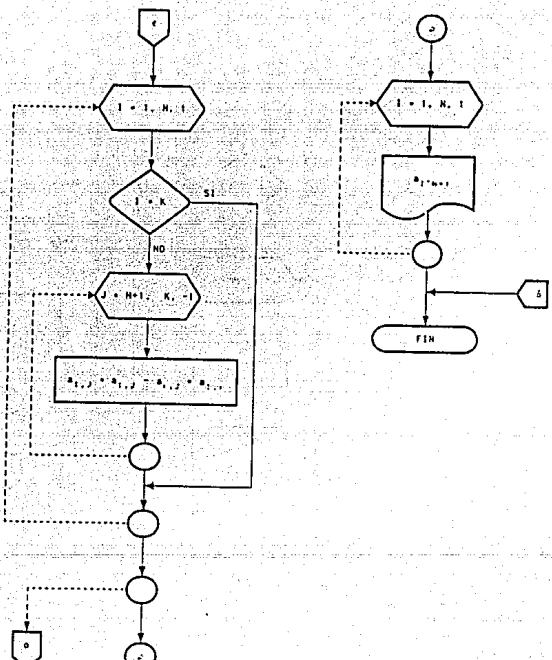
Paso 6 Salida (para i = 1, ..., n: Imprimir xc,)

Terminar.

METODO DE GAUSS-JORDAN




A
7
$a_{j,k} <> 0$
SI
NO
j <> k
NO
SI
7
ES UNA MATRIZ SINGULAR
I = K, N+1, 1
AA = $a_{j,i}$
$a_{j,i} = a_{k,i}$
$a_{k,i} = AA$
J = N+1, K, -1
$a_{k,J} = \frac{a_{k,J}}{a_{k,k}}$
C


C
I = I, N, 1
I = K
SI
NO
J = N+1, K, -1
FIN

```
Program Metodo_Gauss_Jordan;
uses Crt;

var a,b : array[1..30,1..30] of real;
    x : array[1..30] of real;
    mensaje, i, j, k, n, y, en : integer;
    AA, z : real;
    c : char;

procedure Inicializa;
begin
  mensaje := 0;
  i := 0;
  j := 0;
  k := 0;
  n := 0;
  y := 0;
  en := 0;
  AA := 0;
  z := 0;
  c := chart(0);
  for i := 1 to 30 do
   begin
    x[i] := 0;
   end;
end;

procedure Captura;
begin
  clrscr;
  gotoxy(33,3);writet('GAUSS - JORDAN');
  gotoxy(18,5);writet('Resolución de sistemas de ecuaciones lineales');
  gotoxy(29,8);writet('Orden del sistema: n = ');
  gotoxy(52,8);read(n);
  z := (80 - n*10)/2 - 2*(n-1);
  y := trunc(z);
  for i := 1 to n do
    for j := 1 to n + 1 do
      begin
        gotoxy(y + 12*(j-1)-3,10+i*2);writet('A[',i,',',j,'] = ');
      end;
  for i := 1 to n do
    for j := 1 to n + 1 do
      begin
        gotoxy(y + 12*(j-1)+6,10+i*2);read(a[i,j]);
      end;
  b := a;
  gotoxy(20,24);writet('Presione cualquier tecla para continuar');
  gotoxy(60,24);c := readkey;
end;
```

```
procedure Proceso;
begin
    for j := n+1 downto k do
      begin
        a[k,j] := a[k,j] / a[k,k];
      end;
    for i := 1 to n do
      begin
       if i <> k then
                 begin
                   for j := n+1 downto k do
                     begin
                       a[i,j] := a[i,j] - a[k,j]*a[i,k];
                     end;
                 end;
      end;
end;

procedure Intercambio;
begin
     for j := k to n do
       if a[j,k] <> 0 then
                   begin
                     if j <> k then
                                 begin
                                   for i := k to n+1 do
                                     begin
                                       AA := a[j,i];
                                       a[j,i] := a[k,i];
                                       a[k,i] := AA;
                                     end;
                                  proceso;
                                 end
                               else
                                 proceso;
                   end;
end;

procedure GaussJordan;
begin
  for k := 1 to n do
    begin
      Intercambio;
    end;
end;

procedure Resultado;
begin
  clrscr;
  gotoxy(33,3);writeln'GAUSS - JORDAN');
```

```
  gotoxy(18,5);
  writeln('Resolución de sistemas de ecuaciones lineales');
  gotoxy(29,7);writeln('Orden del sistema: n = ',n);
  z := (80 - n*10)/2 - 2*(n-1);
  y := trunc(z);
  for i := 1 to n do
    for j := 1 to n+1 do
      begin
        gotoxy(y + 12*(j-1)-3,7+i*2);write('A[',i,',',j,'] = ');
      end;
  for i := 1 to n do
    for j := 1 to n+1 do
      begin
        gotoxy(y + 12*(j-1) + 6,7 + i*2);write(A[i,j]:2:0);
      end;
  if mensaje = 1 then
    begin
      gotoxy(25,7 + n*2 + 3);write('Se trata de una matriz singular');
    end
  else
    begin
      gotoxy(27,7 + n*2 + 7);write('La solución del sistema es:');
      for i := 1 to n do
        begin
          gotoxy((y-n) + 17*(i-1),7 + n*2 + 9);
          write('X',i,' = ',A[i,n+1]:4:6);
        end;
    end;
  gotoxy(20,24);writeln('Presione cualquier tecla para continuar');
  gotoxy(60,24);c := readkey;
end;


begin
  Inicializa;
  Captura;
  GaussJordan;
  Resultado;
end.
```

MATRIZ TRIDIAGONAL

Sistemas de ecuaciones de la forma especial siguiente, siendo A igual

$$\begin{bmatrix} a_{11}x_1 + a_{12}x_2 & & & & = a_{1,n+1} \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 & & & & = a_{2,n+1} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ & a_{n-1,n-2}x_{n-2} + a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n & & & = a_{n-1,n+1} \\ & & a_{n,n-1}x_{n-1} + a_{n,n}x_n & & = a_{n,n+1} \end{bmatrix}$$

ejemplifica a la matriz de banda de h x h, teniendo p y q, como enteros, siendo $1 < p, q < n$, con la propiedad de que $a_{ij} = 0$, siempre que $i + p <= q$ ó $j + q <= i$. El ancho de banda será dado por $w = p + q - 1$. Existen dos casos particulares de estas matrices, cuando:

p y q igual a 2 $\qquad w = 3$

p y q igual a 4 $\qquad w = 7$

en el primer caso el ancho de banda define a la matriz tridiagonal utilizada en varias aplicaciones como los trazadores cúbicos y el segundo utilizado para la resolución de problemas de valor en la frontera.

La ocurrencia de la matriz tridiagonal es frecuente en la práctica, para su solución usaremos el método de la reducción de Crout, el actual mantiene la propiedad de que los elementos diferentes de cero están alrededor de la diagonal principal. Los algoritmos de

factorización se simplifican para la cantidad de elementos cero factorizando la matriz tridiagonal.

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & \cdots & 0 \\ a_{21} & a_{22} & a_{23} & \cdots & 0 \\ 0 & \cdot & \cdot & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdots & a_{n-1n} \\ 0 & 0 & 0 & a_{nn-1} & a_{nn} \end{bmatrix}$$

en

$$L = \begin{bmatrix} L_{11} & 0 & \cdot & \cdots & 0 \\ L_{21} & L_{22} & \cdot & \cdots & \cdot \\ 0 & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & 0 \\ 0 & \cdot & 0 & L_{nn-1} & L_{nn} \end{bmatrix} \qquad U = \begin{bmatrix} 1 & U_{12} & 0 & \cdots & 0 \\ 0 & 1 & \cdot & \cdots & \cdot \\ 0 & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & U_{n-1n} \\ 0 & \cdot & 0 & U & 1 \end{bmatrix}$$

En virtud del hecho que los elementos tanto sobre como bajo de la diagonal no son cero, es fácilmente verificable la correspondencia entre los elementos de la matriz A y los correspondientes elementos de la matriz L y U, tomando una forma simplificada

$$\begin{aligned} a_{i,i-1} &= L_{i,i-1} && (\ i = 2,\ 3,\ \ldots,\ n\ ); \\ a_{i,i} &= L_{i,i-1}\,U_{i-1,i} + L_{i,i} && (\ i = 2,\ 3,\ \ldots,\ n\ ); \\ a_{i,i} &= L_{i,i}\,U_{i,i+1} && (\ i = 1,\ 2,\ \ldots,\ n\ ). \end{aligned}$$

para solucionar este sistema de ecuaciones, encontramos los términos diferentes de cero fuera de la diagonal de L con la segunda ecuación, para después obtener alternadamente los elementos restantes en U y L con las dos últimas ecuaciones, las cuales se almacenarán en los elementos correspondientes de la matriz original A.

Algoritmo:

Para resolver el sistema lineal de n x n, el cual se supone tiene solución única.

Entrada: El número de ecuaciones n; los elementos de la matriz aumentada A.

Salida: La solución $x_1, \ldots, x_n$.

Paso 1 Tomar $\beta_1 = b_1$

Paso 2 Tomar $\gamma_1 = d_1 / \beta_1$

Paso 3 Para i = 2, ..., n hacer

$$\beta_i = b_i - \frac{a_i * c_i}{\beta_{i-1}}, \qquad \alpha_i = \frac{d_i - a_i * \gamma_{i-1}}{\beta_i}$$

Paso 4 Tomar $x_n = \gamma_n$

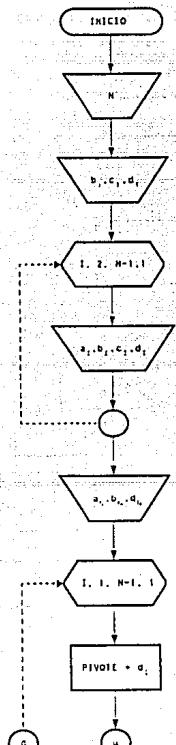
Paso 5 Para i = n-1, ..., 1 decrementado en 1

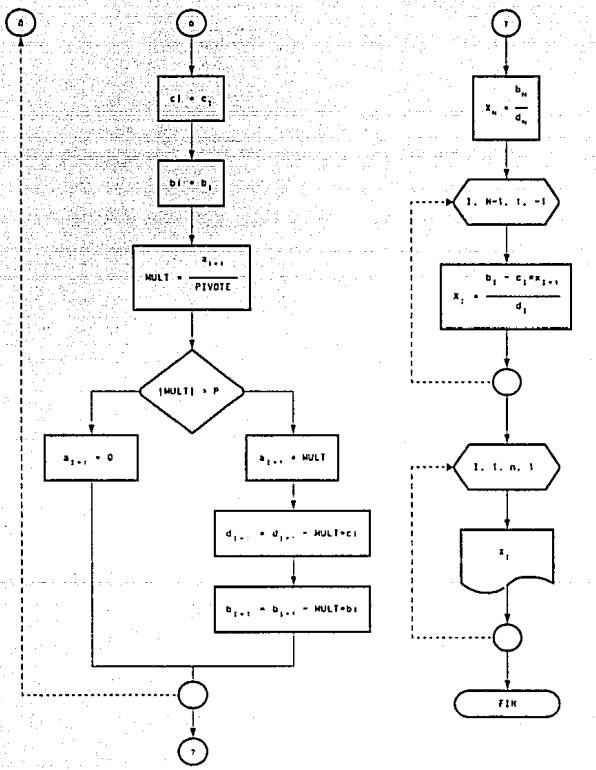
$$x_i = \gamma_i - \frac{c_i * x_{i+1}}{\beta_i}$$

Paso 6 Salida (para i = 1, ..., n; Imprimir x_i)

Terminar.

MATRIZ TRIDIAGONAL




A
6
7
ci = c_i
bi = b_i
MULT = $\frac{a_{i,k}}{PIVOTE}$
|MULT| > P
$a_{i,k} = 0$
$a_{i,k} = MULT$
$d_{i,k} = d_{i,k} - MULT*ci$
$a_{i+1} = b_{i,k} - MULT*bi$
7
$X_n = \frac{b_n}{d_n}$
I, N-1, 1, -1
$X_i = \frac{b_i - c_i*X_{i+1}}{d_i}$
I, 1, n, 1
X_i
FIN

```
Program Matriz_Tridiagonal;
uses crt;
var x, a, b, c, d, beta, gama : array [1..20] of real;
   i, j, n, col : Integer;
    cont : char;

procedure Inicializa;
begin
 i := 1;
 j := 0;
 n := 0;
 col := 0;
 cont := chr(0);
end;

procedure Captura;
begin
 clrscr;
 gotoxy(31,2);write('MATRIZ  TRIDIAGONAL');
 gotoxy(25,5);write('Suministre los siguientes datos');
 gotoxy(27,7);write('Número de ecuaciones: n  = ');
 gotoxy(31,9);write('Matriz  Tridiagonal');
 gotoxy(53,7);read(n);
 col := 80-(n-2)*10;
 gotoxy(col,11);write('b[',i,']');
 gotoxy(col+10,11);write('c[',i,']');
 gotoxy(col+40,11);write('d[',i,']');
 for i := 2 to n-1 do
   begin
     gotoxy(col-j,10+i);write('a[',i,']');
     gotoxy(col+j+10,10+i);write('b[',i,']');
     gotoxy(col+j+20,10+i);write('c[',i,']');
     gotoxy(col+40,10+i);write('d[',i,']');
     j := j+10;
   end;
 gotoxy(col+20,10+n);write('a[',n,']');
 gotoxy(col+30,10+n);write('b[',n,']');
 gotoxy(col+40,10+n);write('d[',n,']');
 gotoxy(col+5,11);read(b[1]);
 gotoxy(col+15,11);read(c[1]);
 gotoxy(col+45,11);read(d[1]);
 j := 0;
 for i := 2 to n-1 do
   begin
     gotoxy(col+j+5,10+i);read(a[i]);
     gotoxy(col+j+15,10+i);read(b[i]);
     gotoxy(col+j+25,10+i);read(c[i]);
     gotoxy(col+45,10+i);read(d[i]);
     j := j+10;
   end;
```

```
    gotoxy(col+25,10+n);read(a[n]);
    gotoxy(col+35,10+n);read(b[n]);
    gotoxy(col+45,10+n);read(d[n]);
    gotoxy(21,24);writet('Para continuar presione cualquier tecla');
    gotoxy(60,24);cont := readkey;
  end;

  procedure MatrizTridiagonal;
  begin
    beta[1] := b[1];
    gama[1] := d[1]/beta[1];
    for i := 2 to n do
      begin
        beta[i] := b[i] - (a[i]*c[i-1])/beta[i-1];
        gama[i] := (d[i] - a[i]*gama[i-1])/beta[i];
      end;
    x[n] := gama[n];
    for i := n-1 downto 1 do
      begin
        x[i] := gama[i] - (c[i]*x[i+1])/beta[i];
      end;
  end;

  procedure Resultado;
  begin
    for i := 1 to n do
      begin
        gotoxy(25,12+n+i);writet('x[',i,'] = ',x[i]:5:10);
      end;
    gotoxy(60,24);cont := readkey;
  end;

  begin
    Inicializa;
    Captura;
    MatrizTridiagonal;
    Resultado;
  end.
```

SISTEMAS DE ECUACIONES NO LINEALES

En esta sección se verán de los métodos para solucionar sistemas de ecuaciones no lineales (algebraicas o trascendentes). Dichos sistemas surgen frecuentemente en la práctica. Las técnicas que se verán son generalizaciones de algunos de los métodos aplicados a la solución de una sola ecuación no lineal.

Estos métodos son en principio, aplicaciones a sistemas de n ecuaciones con n incognitas.

$$f_k(x_1, x_2, \ldots, x_n) = 0 \qquad k = 1, 2, \ldots, n$$

en el caso que n = 2 tendremos un sistema definido por dos ecuaciones y dos incógnitas

$$y = f_2(x,y)$$

la generalización a n ecuaciones y n incógnitas deberá ser directa una vez que han sido captadas las ideas fundamentales. Ahora, asumiendo la solución al sistema anterior en forma vectorial tendremos

$$f(x) = 0 \qquad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad f(x) = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix}$$

La solución del sistema se puede considerar como un proceso de dos pasos; primero encontrar las curvas solución en el plano (x_1,x_2). $z_1 = f_1(x_1,x_2)$ y $z_2 = f_2(x_1,x_2)$ y segundo encontrar los puntos de intersección de estas curvas solución en el plano (x_1,x_2).

En principio, el método de Muller y los métodos de Newton de alto orden extendidos son buenos, pero la cantidad de trabajo a desarrollar se incrementa rápidamente con el número de ecuaciones y por esto su uso se hace limitado.

Los principales métodos para sistemas de ecuaciones no lineales están basados en modelos lineales tales como el método de Newton-Raphson o el método de la secante.

MÉTODO DE NEWTON

Consideraremos el problema de la solución de sistemas de ecuaciones, de manera similar a el método de Newton-Raphson para una ecuación. Hay más de una manera de ver y derivar el método de Newton-Raphson para resolver un sistema de ecuaciones no lineales. Aquí, comenzaremos una derivación analítica, y entonces daremos una perspectiva geométrica.

Aplicamos el teorema de Taylor para funciones de dos variables para cada una de las ecuaciones $f_i(x_1, x_2) = 0$, expandiendo $f_i(\alpha)$ alrededor de x_{α}; para $i = 1, 2$

$$0 = f_i(\alpha) = f_i(x_0) + (\alpha_1, x_{1,0}) \frac{\partial f_i(x_0)}{\partial x_1} + (\alpha_2 - x_{2,0}) \frac{\partial f_i(x_0)}{\partial x_2} + \frac{1}{2}\left[(\alpha_1 - x_{1,0}) \frac{\partial}{\partial x_1} + (\alpha_2 - x_2) \frac{\partial}{\partial x_2}\right]^2 f_i(\xi^{(i)})$$

con $\xi^{(i)}$ en el segmento de línea entre x_{α} y α. Si omitimos los términos de segundo orden, obtendremos la siguiente aproximación

$$0 = f_1(x_0) + (\alpha_1, x_{1,0}) \frac{\partial f_1(x_0)}{\partial x_1} + (\alpha_2 - x_{2,0}) \frac{\partial f_1(x_0)}{\partial x_2}$$
$$0 = f_2(x_0) + (\alpha_1, x_{1,0}) \frac{\partial f_2(x_0)}{\partial x_1} + (\alpha_2 - x_{2,0}) \frac{\partial f_2(x_0)}{\partial x_2}$$

en forma matricial

$$0 = f(x_0) + F(x_0)(\alpha - x_0)$$

con $F(x_n)$ como la matriz de jacobiana de f, dada como

$$F(x_n) = \begin{bmatrix} \frac{\partial f_1(x_n)}{\partial x_1} & \frac{\partial f_1(x_n)}{\partial x_2} \\ \frac{\partial f_2(x_n)}{\partial x_1} & \frac{\partial f_2(x_n)}{\partial x_2} \end{bmatrix}$$

resolviendo para α,

$$\alpha = x_n - F(x_n)^{-1} f(x_n) = x_1$$

La aproximación x1 debe ser un mejoramiento de x_n, x_n es escogida lo suficientemente cercana a α. Esto nos lleva al método iterativo

$$x_{n+1} = x_n - F(x_n)^{-1} f(x_n), \quad n \geq 0$$

Este es el método de Newton para resolver el sistema no lineal $f(x) = 0$.

Actualmente en la práctica no se obtiene la matriz inversa de f(x), particularmente para sistemas de más de dos ecuaciones. En cambio, resolveremos un sistema lineal con un término de corrección para x_n

$$f(x_n)\,\delta_{n+1} = -f(x_n)$$

$$x_{n+1} = x_n + \delta_{n+1}$$

Esto es más eficiente en tiempo de cálculo, requiriendo tantas operaciones como unas tres inversiones de $f(x_n)$.

La derivación geométrica para este método es analoga al método de Newton-Raphson. La gráfica en el espacio de la ecuación.

$$z = f_i(x_o) + (\alpha_1 - x_{1,o}) \frac{\partial f_i(x_o)}{\partial x_1} + (\alpha_2, x_{2,o}) \frac{\partial f_i(x_o)}{\partial x_2} \equiv P_i(x_1, x_2)$$

es un plano que es tangente a la gráfica de $z = f_i(x_1,x_2)$ en el punto x_o para $i = 1, 2$. Si x_o es cercano a α, entonces estos planos tangentes deben ser buenas aproximaciones a las superficies asociadas de $z = f(x_1,x_2)$, para $x = (x_1,x_2)$ cercana a α. Entonces la intersección de las curvas cero de los planos tangentes $z = P_i(x_1,x_2)$ deben ser una buena aproximación a la intersección correspondiente α de las curvas cero de las superficies originales $z = f_i(x_1,x_2)$. Esto resulta en la declaración de la expansión de Taylor hasta el segundo término de las dos ecuaciones no lineales. La intersección de las curvas cero de $z = f_i(x_1,x_2)$, es $i = 1, 2$, es el punto x_1.

MÉTODO DE LA SECANTE

El método de la secante para sistemas de ecuaciones se obtiene por una extensión directa de la idea de una ecuación. Sin embargo, esto no es tan fácil como para representarlo en fórmulas, y de tal manera describimos esto en palabras. En el k-ésimo paso de iteración habremos guardado x_k, x_{k+1}, ..., x_{k+n}. Para cada función $f_i(x)$ determinamos el plano que interpola $f_i(x)$ en $f_i(x_{k+j})$ para $j = 1, 2, \ldots, n$. así los n planos son calculados para un modelo de n componentes de f(x); cada plano requiere la solución de $n+1$ ecuaciones lineales con $n+1$ variables. Entonces se resuelve el sistema lineal de $n \times n$ obtenido al colocar estas funciones lineales iguales a cero simultáneamente, esto es, se localiza dónde se intersectan los planos simultáneamente en cero. La solución de este sistema es la siguiente iteración x_{k+1}. Tanto este método como el de Newton-Raphson requieren resolver n ecuaciones en el último paso de cada iteración. Esto significa una gran cantidad de trabajo, pero esto es inherente utilizando el modelo lineal.

CAPITULO VII

AJUSTE

DE DATOS

AJUSTE DE DATOS

El análisis de muchos problemas en ingeniería química consiste en la utilización de datos experimentales para verificar teorías o desarrollar fórmulas empíricas. Con frecuencia algunos de los datos puntuales son inexactos y deben encontrarse métodos que permitan elegir información adecuada con razonable certeza.

Se puede adelantar mucho en el análisis de los resultados o datos representándolos gráficamente, eligiendo las coordenadas adecuadas e inspeccionándolos visualmente.

Los resultados experimentales pueden utilizarse también para determinar un valor medio de una variable, o un conjunto de datos puede integrarse para encontrar un valor total. La determinación de una velocidad media a través de una sección transversal de un tubo o pieza de una instalación sería un ejemplo del primer caso; y la predicción de una composición acumulada a partir de un conjunto de valores de composiciones puntuales sería un ejemplo del segundo caso.

POLINOMIO DE INTERPOLACIÓN

La interpolación polinomial se usa cuando los datos son exactos. Suponiendo que los valores de x_i en un conjunto de datos (x_i,y_i) dado, donde $i = 1, 2, 3, \ldots, n$, posiblemente equidistantes o espaciados en donde la interpolación lineal no tiene la aproximación suficiente para nuestros propósitos. En tales casos tomamos en cuenta una herramienta básica para encontrar fórmulas, este método es el de coeficientes indeterminados. Este supone la forma de la respuesta, y las condiciones en la fórmula determinan la forma de los coeficientes arbitrarios, es decir, determina un polinomio que pase por los puntos dados. Por ejemplo, dados $n+1$ puntos (x_i,y_i), $i = 1, 2, \ldots, n+1$ podemos accesar un polinomio de grado n

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n = \sum_{k=0}^{n} a_k x^k$$

para los demás datos (la forma del polinomio tiene $n+1$ coeficientes para ser determinados por las $n+1$ condiciones) la condición de que el polinomio $P_n(x)$ pase por el conjunto de puntos (x_i,y_i) es

$$y_i = P_n(x_i) = \sum_{k=0}^{n} a_k x_i^k, \qquad i = 1, 2, \ldots, n+1$$

las $n+1$ ecuaciones determinaran a los coeficientes a_i.

El determinante de las $n+1$ ecuaciones originado al evaluar en cada punto el polinomio

de grado n e igualar este resultado al de la función dada para este punto, se conoce como determinante de Vandermonde

$$V_{n+1} = \begin{vmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{n+1} & x_{n+1}^2 & \cdots & x_{n+1}^n \end{vmatrix} = |x_i^{i-1}|$$

teniendo en cuenta que el determinante no podrá ser cero si $x_i <> x_j$ para $i <> j$; a partir de esto, siempre podremos resolver para el a_k y tendremos una solución a el problema de interpolación para polinomios.

Si resolvemos para el a_k usando determinantes, substituimos el resultado en el polinomio y finalmente rearreglamos adecuadamente, obtendremos

$$\begin{vmatrix} y & 1 & x_1 & x_1^2 & \cdots & x_1^n \\ y_1 & 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ y_n & 1 & x_{n+1} & x_{n+1}^2 & \cdots & x_{n+1}^n \end{vmatrix} = 0$$

Podemos ver directamente que ésta es la solución, expandiendo a los elementos del renglón superior, se trata obviamente un polinomio de grado n. Si x e y toman los valores de x_i e y_i, entonces dos renglones deben ser idénticos, y de aquí que el determinante deberá ser cero. Esta observación muestra que el polinomio pasa a través de los puntos dados.

Algoritmo:

Para evaluar los coeficientes del polinomio de interpolación P en los (n+1) puntos distintos $x_0, \ldots, x_n$ para la función f:

Entrada: Modelo del polinomio

1.- $a_0 + a_1x_i + a_2x_i^2 + \ldots$

2.- $a_0 + a_1x_{i_1} + a_2x_{i_2} + \ldots$

número de datos n; datos x_i ó x_{i_n} dependiendo del modelo.

Salida: Los coeficientes del polinomio.

Paso 1 Si m = 1 entonces realizar paso 2 si no, paso 3

Paso 2 Para i = 1, ..., n

Para j = 1, ..., n

Tomar $A_{ij} = x_i^{j-1}$

Tomar $a_{i,n+1} = y_i$

Paso 3 Para i = 1, ..., n

Para j = 2, ..., n

Tomar $A_{ij} = x_{i_j}$

Tomar $a_{i1} = 1$;

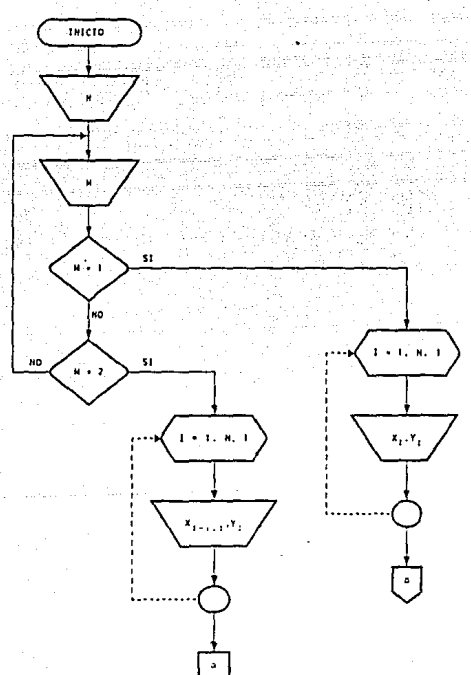
Tomar $a_{i,n+1} = y_i$

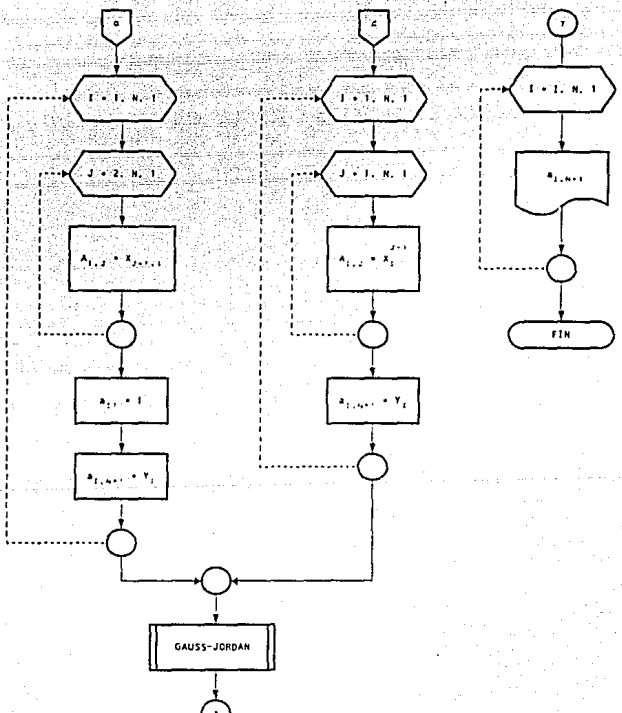
Paso 4 Resolver sistema con Gauss-Jordan

Paso 5 Salida (Para i = 1, ..., n+1; Imprimir $a_{i,n+1}$)

Terminar.

POLINOMIO DE INTERPOLACION




I = 1, N, 1
J = 2, N, 1
$A_{I,J} = X_{J-1,I}$
$A_{I,1} = 1$
$A_{I,N+1} = Y_I$
I = 1, N, 1
J = 1, N, 1
$A_{I,J} = X_I^{J-1}$
$A_{I,N+1} = Y_I$
GAUSS-JORDAN
I = 1, N, 1
$a_{I,N+1}$
FIN

```
Program Polinomio_de_Interpolacion;
uses crt;

var i, j, k, n, m, v, pot  : Integer;
    arg, valor, AA : real;
    x, y : array [1..20] of real;
    xx : array [1..20,1..20] of real;
    a : array [1..50,1..51] of real;
    c : char;
procedure Inicializa;
begin
 i := 0;
 j := 0;
 k := 0;
 n := 0;
 m := 0;
 v := 0;
 pot := 0;
 arg := 0;
 valor := 0;
 AA := 0;
 c := char(0);
 for i := 1 to 20 do
   for j := 1 to 20 do
     xx[i,j] := 0;
 for i := 1 to 50 do
   for j := 1 to 51 do
     a[i,j] := 0;
 for i := 1 to 20 do
  begin
   x[i] := 0;
   y[i] := 0;
  end;
end;

procedure Captura;
procedure Entrada1;
begin
   Gotoxy(30,6);Writeln('Número de datos n = ');
   Gotoxy(21,24);Writeln('Presione cualquier tecla para continuar');
   Gotoxy(50,6);Readln(n);
   For i := 1 to n do
     begin
       Gotoxy(26,7+i);Writeln('X[',i,'] = ');
       Gotoxy(33,7+i);Readln(x[i]);
       Gotoxy(46,7+i);Writeln('Y[',i,'] = ');
       Gotoxy(53,7+i);Readln(y[i]);
     end;
   Gotoxy(60,24);c := readkey;
end;

procedure Entrada2;
begin
   Gotoxy(30,6);Writeln('Número de datos n = ');
   Gotoxy(21,8);Writeln('Número de variables independientes = ');
   Gotoxy(21,24);Writeln('Presione cualquier tecla para continuar');
```

```
    Gotoxy(50,6);Readln(n);
    Gotoxy(58,8);Readln(v);
    For i := 1 to n do
     begin
       Gotoxy(26,9+i);Writeln('Y[',i,'] = ');
       Gotoxy(33,9+i);Readln(Y[i]);
     end;
    For i := 1 to v do
    begin
     For j := 1 to n do
       begin
         Gotoxy(43,9+j);Clreol;
       end;
     For j := 1 to n do
       begin
         Gotoxy(45,9+j);Writeln('X[',i,',',j,'] = ');
         Gotoxy(54,9+j);Readln(XX[i,j]);
       end;
    end;
    Gotoxy(60,24);c := readkey;
end;

begin
    Clrscr;
    Gotoxy(27,5);Writeln('POLINOMIO DE INTERPOLACION');
    Gotoxy(26,8);Writeln('Seleccione el modelo deseado:');
    Gotoxy(23,12);Writeln('1- Y = A0 + A1Xi - A2Xi^2 + A3Xi^3');
    Gotoxy(23,16);Writeln('2- Y = A0 + A1X1 - A2X2i + A3X3i');
    Gotoxy(36,23);Writeln('Modelo:[_]');
    Gotoxy(44,23);Readln(M);
    if (M < 1) and (M > 2) then Captura;
    Clrscr;
    Gotoxy(27,3);Writeln('POLINOMIO DE INTERPOLACION');
    if M = 1 then Entrada1
            else Entrada2;
end;


procedure Proceso;
procedure ProcesoGJ;
begin
    for j := n+1 downto k do
      begin
        a[k,j] := a[k,j] / a[k,k];
      end;
    for i := 1 to n do
      begin
       if i <> k then
                begin
                  for j := n+1 downto k do
                    begin
                      a[i,j] := a[i,j] - a[k,j]*a[i,k];
                    end;
                end;
      end;
end;
```

```
procedure Intercambio;
begin
    for j := k to n do
      if a[j,k] <> 0 then
                  begin
                    if j <> k then
                                begin
                                  for i := k to n+1 do
                                    begin
                                      AA := a[j,i];
                                      a[j,i] := a[k,i];
                                      a[k,i] := AA;
                                    end;
                                  ProcesoGJ;
                                end
                              else
                                ProcesoGJ;
                  end;
end;

procedure GaussJordan;
begin
 for k := 1 to n do
  begin
    Intercambio;
  end;
end;

function xe(arg : real; pot : integer) : real;
var v : real;
    sign : integer;
begin
   if pot = 0 then v := 1
      else
        if arg > 0 then v := Exp(pot*Ln(arg))
          else
            begin
              sign := -1;
              if odd(abs(pot)) then
                  v := sign*Exp(pot*Ln(abs(arg)))
                else
                  v := Exp(pot*Ln(abs(arg)));
            end;
    xe := v
end;

procedure ProcesoA;
begin
   For i := 1 to n do
    begin
      For j:= 1 to n do
        a[i,j] := xe(X[i],j-1);
      a[i,n+1] := Y[i];
    end;
end;
```

```
procedure ProcesoB;
begin
  clrscr;
  For i := 1 to n do
    begin
      For j := 1 to n do
        a[i,j] := xx[j-1,i];
      a[i,1] := 1;
      a[i,n+1] := y[i];
    end;
end;

begin
  if m = 1 then
            begin
              ProcesoA;
              GaussJordan;
            end
          else
            begin
              ProcesoB;
              GaussJordan;
            end;
end;

procedure Resultado;
begin
  clrscr;
  gotoxy(27,3);writeln('POLINOMIO DE INTERPOLACION');
  gotoxy(22,5);writeln('Los coeficientes de la ecuacion son:');
  gotoxy(20,24);writeln('Presione cualquier tecla para continuar');
  for i := 1 to n do
    begin
      gotoxy(29,7+i);
      writeln('A',i-1,' = ',A[i,n+1]);
    end;
  gotoxy(60,24);c := readkey;
end;


begin
  Captura;
  Proceso;
  Resultado;
end.
```

POLINOMIO FUNDAMENTAL DE NEWTON

La fórmula general de interpolación de Newton puede ser derivada iniciando con algún par supuesto de valores de x e y, y los pares de valores dados, escribiendo abajo las diferencias divididas en orden ascendente, y entonces resolver para y en pasos sucesivos hasta que se encuentren tantos términos como se desee.

Definiendo el concepto de derivada:

$$\left.\frac{df(x)}{dx}\right|_{x_o} = f'(x_o) = \lim_{x \to x_o} \frac{f(x) - f(x_o)}{x - x_o}$$

tomando la definición

$$f[x, x_o] = \frac{f(x) - f(x_o)}{x - x_o}, \quad x \neq x_o$$

en la que $f[x,x_o]$ se denomina primer cociente incremental finito o finito de primer orden respecto de los argumentos x, x_o.

Con el fin de permitir aproximaciones análogas para derivadas de oreden superior, el concepto de cociente incremental se amplia de la forma indicada en la siguiente tabla

ORDEN	DIFERENCIA	
0	$f[x_0]$	$f(x_0)$
1	$f[x_1, x_0]$	$\dfrac{f[x_1] - f[x_0]}{x_1 - x_0}$
2	$f[x_2, x_1, x_0]$	$\dfrac{f[x_2, x_1] - f[x_1, x_0]}{x_2 - x_0}$
.	.	.
.	.	.
.	.	.
n	$f[x_n, \ldots, x_0]$	$\dfrac{f[x_n, \ldots, x_1] - f[x_{n-1}, \ldots, x_0]}{x_n - x_0}$

Definiendo al cociente incremental de orden n como:

$$f[x_n, \ldots, x_0] = \sum_{i=0}^{n} \frac{f(x_i)}{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)}$$

Donde los n+1 puntos base esten definidos como sigue:

$$x_0 = f(x_0)$$

$$x_1 = f(x_1)$$

.

.

.

$$x_n = f(x_n)$$

Para establecer el polinomio de interpolación de f(x), en el intervalo $x_0 < x < x_n$ tendremos n+1 puntos base. Al conformar a f(x) con los coeficientes incrementales y para considerar el error de truncación se incluye el término R(x), obteniendo lo siguiente

$$f(x) = f(x_0) + (x-x_0) f[x_1, x_0] + (x-x_0)(x-x_1) f[x_2, x_1, x_0] + \ldots + (x-x_0) \ldots (x-x_{n-1}) f[x_n, \ldots, x_1, x_0] + R(x)$$
$$= P_n(x) + R(x)$$

Donde R(x), como se estableció, es la discrepancia entre f(x) y P(x), cuando el polinomio P(x) pasa por los puntos $(x_0, f(x_0))$, $(x_1, f(x_1))$, ..., $(x_n, f(x_n))$, siendo así el error de la aproximación.

Algoritmo:

Para obtener los coeficientes de diferencia dividida del polinomio de interpolación P en los (n+1) puntos distintos $x_0, \ldots, x_n$ para la función f:

Entrada: número de puntos n; los valores de $x_0, \ldots, x_n$ y de $f(x_0), \ldots, f(x_n)$.

Salida: Los coeficientes del polinomio interpolante $f_{0,0}$.

Paso 1 Para i = 0 hacer

Tomar $f_{i,i} = f(x_i)$

Paso 2 Para k = 1, ..., n-1 realizar

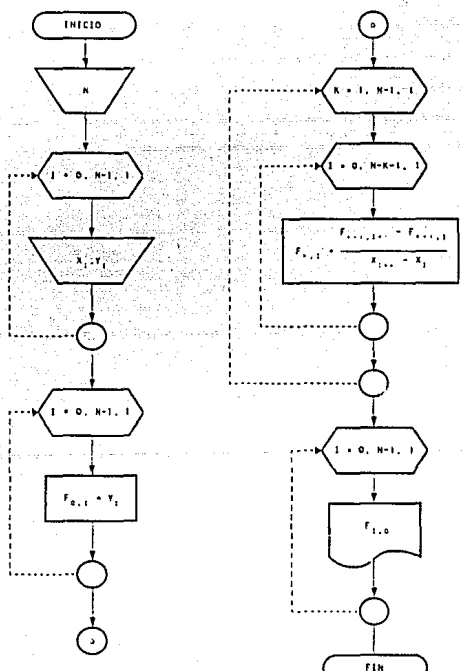
Para i = 0, ..., n-k-1 hacer

$$f_{k,i} = \frac{f_{i+1,\ldots,i+k} - f_{i,\ldots,i+k-1}}{x_{i+k} - x_i}$$

Paso 3 Salida (para i = 0, ..., n-1; imprimir $f_{i,0}$)

Terminar.

POLINOMIO FUNDAMENTAL DE NEWTON



```
Program Polinomio_Fundamental_de_Newton;
uses crt;
var x, fx: array [0..20] of real;
    f : array [0..20,0..20] of real;
    i, j, n : integer;
    c : char;

procedure Inicializa;
begin
  i := 0;
  j := 0;
  n := 0;
  c := char(0);
  for i := 0 to 20 do
    begin
      x[i] := 0;
      fx[i]:= 0;
    end;
  for i := 0 to 20 do
    for j := 0 to 20 do
      f[i,j] := 0;
end;

procedure Captura;
begin
  clrscr;
  gotoxy(25,4);writeln('POLINOMIO FUNDAMENTAL DE NEWTON');
  gotoxy(25,7);writeln('Suministre los siguientes datos');
  gotoxy(29,10);write('Número de datos:  n = ');
  gotoxy(51,10);read(n);
  gotoxy(21,24);write('Presione cualquier tecla para continuar');
  gotoxy(21,60);c:=readkey;
  clrscr;
  gotoxy(1,10);clreol;
  gotoxy(25,2);writeln('Suministre los siguientes datos');
  gotoxy(30,4);writeln('x          f(x)');
  for i := 0 to n-1 do
    begin
      gotoxy(30,6+i);read(x[i]);
      gotoxy(45,6+i);read(fx[i]);
    end;
  gotoxy(21,24);write('Presione cualquier tecla para continuar');
  gotoxy(21,60);c:=readkey;
end;

procedure PFN;
  begin
  clrscr;
    for i := 0 to n-1 do
      f[0,i] := fx[i];
```

```
    for i := 1 to n-1 do
      for j := 0 to n-i-1 do
        d[i,j] := (d[i-1,j+1] - d[i-1,j]) / (x[j+i] - x[j]);
  end;

procedure Resultado;
begin
  clrscr;
  gotoxy(25,3);write('POLINOMIO FUNDAMENTAL DE NEWTON');
  gotoxy(14,6);
  write('Los coeficientes del polinomio son los siguientes');
  for i := 0 to n-1 do
    begin
      gotoxy(23,8+i);write('Coeficiente ',i,' = ',d[i,0]:20:8);
    end;
  gotoxy(21,24);write('Presione cualquier tecla para continuar');
  gotoxy(21,60);c:=readkey;
end;

begin
  Inicializa;
  Captura;
  PFN;
  Resultado;
end.
```

MÍNIMOS CUADRADOS NO LINEALES

El método de mínimos cuadrados es utilizado en situaciones donde hay muchos más datos disponibles o condiciones a satisfacer que parámetros a ajustar. En la aproximación por mínimos cuadrados se utilizan con más frecuencia polinomios que modelos lineales, aún cuando en ocasiones alguna familia de rectas puede dar una buena aproximción. En la práctica intentamos cuantificar alguna cantidad x y realizar m medidas x_i.

En el gráfico de la siguiente página no se observa a x, solamente las medidas x_i con errores de medición ϵ_i, esto es, nosotros vemos:

$$x_i = x + \epsilon_i \quad i = 1, 2, \ldots, n$$

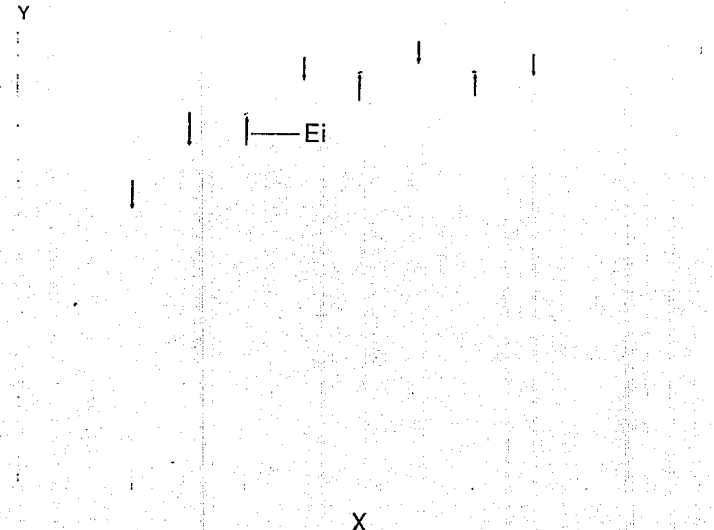
viendo a ϵ_i como "ruido", y a x como el valor verdadero.

El principio de la condición de los mínimos cuadrados consiste en el mejor estimado de x siendo ese número el que minimiza la suma de los cuadrados de las desviaciones de el dato calculado respecto al dato real.

$$f(x) = \sum_{i=1}^{m} \epsilon_i^2 = \sum_{i=1}^{n} (x_i - x)^2$$

$$\sum_{i=1}^{m} (f_{obs} - f_{aprox})^2 \qquad minimiza$$

MINIMOS CUADRADOS NO LINEALES



En el análisis final, la utilidad de este polinomio depende qué tan útil sea el resultado obtenido en la práctica y qué tan fácilmente pueda ser usado.

Este principio es equivalente a la suposición de que el promedio es el mejor estimado.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

El seleccionar el promedio como el mejor valor es muy común en la práctica en muchas situaciones. No obstante, es inapropiado en algunas otras. Probablemente la mayor falla con este método es que una medición sencilla muy mala puede distorcionar grandemente los resultados, ya que al elevar las desviaciones al cuadrado, dicha medición nos dará un termino que tendrá una parte dominante. Por ello debe tenerse mucho cuidado al utilizar este método, siempre se deberá revisar si alguna o algunas mediciones se disparan.

Cuando el número de puntos es grande, el polinomio de grado n se podrá representar como

$$y = a_0 + a_1x + \ldots + a_nx^n$$

y el representar los datos exactamente por este polinomio, no solamente resulta laborioso y puede ser infructuoso para datos experimentales que contienen invariablemente errores siendo más sensible al tratar de representar los datos aproximadamente por alguna función $y=f(x)$, la cual contiene pocos parámetros, estos parámetros pueden entonces ser determinados de modo

que la curva $y = f(x)$ adecúe los datos "de la mejor manera posible"; el criterio en cuando a que constituye "la mejor manera posible" es arbitrario.

MÍNIMOS CUADRADOS LINEALES

Este método es un caso especial del método de mínimos cuadrados ya que lo que busca es encontrar una línea recta o mejor dicho la recta que mejor se aproxime a los datos, entonces tenemos

$$y(x) = a + bx$$

para un conjunto de datos (x_i, y_i), $i = 1, 2, \ldots, n$. Tenemos dos parámetros a y b y buscamos minimizar la función de dos variables (a y b)

$$f(a, b) = \sum_{i=1}^{m} [y(x_i) - y_i]^2 = \sum_{i=1}^{m} \epsilon_i^2$$

$y(x_i)$ es el valor calculado y y_i es el valor medido. Esto obviamente es la generalización de la ecuación que minimiza la suma de los cuadrados de las desviaciones, tenemos

$$f(a, b) = \sum_{i=1}^{m} (a - bx_i - y_i)^2$$

para minimizar, aplicamos el método de cálculo

$$\frac{\partial f(a, b)}{\partial a} = 2\sum (a - bx_i - y_i) = 0$$
$$\frac{\partial f(a, b)}{\partial b} = 2\sum (a - bx_i - y_i) x_i = 0$$

dividimos por un factor de 2, y reescribiendo, obtenemos

$$aM - b\sum x_i = \sum y_i$$
$$a\sum x_i - b\sum x_i^2 = \sum x_i y_i$$

esta ecuación se resuelve fácilmente para los parámetros a y b. De esta manera obtenemos la línea recta de mínimos cuadrados para los datos.

Algoritmo:

Para evaluar los coeficientes del polinomio de interpolación P en los (n+1) puntos distintos $x_0, \ldots, x_n$ para la función f:

Entrada: Modelo del polinomio

1.- $a_0 + a_1x_1 + a_2x_1^2 + \ldots$

2.- $a_0 + a_1x_{11} + a_2x_{12} + \ldots$

número de datos n; datos x_i ó x_{ij} dependiendo del modelo.

Salida: Los coeficientes del polinomio.

Paso 1 Si m = 1 entonces realizar Paso 2 si no, Paso 3

Paso 2 Para i = 1, ..., n

Para j = 1, ..., n

Tomar $A_{ij} = x_i^{j-1}$

Tomar $a_{i,n+1} = y_i$

Paso 3 Para i = 1, ..., n

Para j = 2, ..., n

Tomar $A_{ij} = x_{i+1}$

Tomar $a_{i1} = 1$;

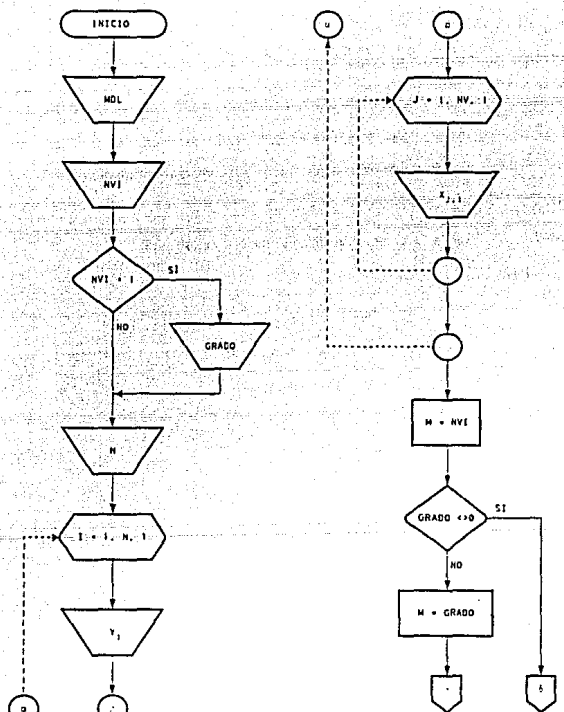
Tomar $a_{i,n+1} = y_i$

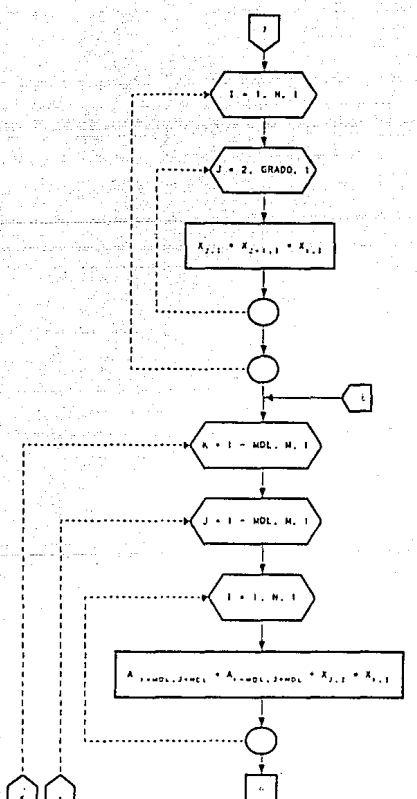
Paso 4 Resolver sistema con Gauss-Jordan

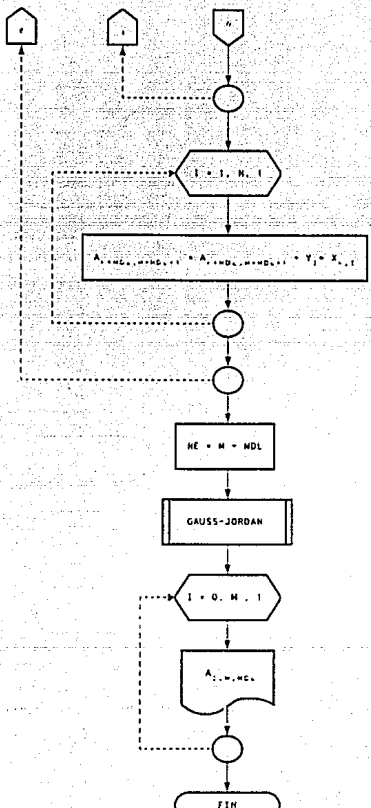
Paso 5 Salida (Para i = 1, ..., n+1: Imprimir $a_{i,n+1}$)

Terminar.

METODO DE MINIMOS CUADRADOS




I = 1, N, 1
J = 2, GRADO, 1
$X_{J,I} = X_{J-1,I} * X_{1,I}$
K = 1 - MDL, M, 1
J = 1 - MDL, M, 1
I = 1, N, 1
$A_{I+MDL,J+MDL} = A_{I+MDL,J+MDL} * X_{J,I} = X_{1,I}$


I = 1, N, 1
A(I+MDL,M+MDL+1) = A(I+MDL,M+MDL+1) * Y(I) * X(I,I)
NE = M + MDL
GAUSS-JORDAN
I = 0, M, 1
A(I,M+MDL)
FIN

```
Program Minimos_Cuadrados_para_Modelos_Lineales;
uses Crt;
const mensaje1 = 'con Ordenada al origen';
      mensaje2 = 'sin Ordenada al origen';
var i, j, k, m, n, en, gp, ne, nv, paso : integer;
    x, y : array [0..20,0..20] of real;
    y : array [1..20] of real;
    Op : Byte;
    mensaje : string[22];
    AA : real;
    c, Tipo : char;

procedure Inicializa;
begin
  i := 0;
  j := 0;
  k := 0;
  m := 0;
  n := 0;  en := 0;
  gp := 0; ne := 0;
  nv := 0; Op := 0;
  AA := 0;
  paso := 0;
  c := chr(0);
  Tipo := chr(0);
  for i := 1 to 20 do
    begin
      y[i] := 0;
      x[0,i] := 1;
    end;
  for i := 1 to 20 do
    for j := 1 to 20 do
      begin
        x[i,j] := 0;
      end;
end;

procedure Desicion;
begin
  gotoxy(44,23);read(Tipo);
  if (Tipo > '2') or (Tipo < '1') then
    begin
      gotoxy(28,21);Textcolor(0);Textbackground(7);
      gotoxy(28,21);write('La opcion debe ser  1 o 2');
      gotoxy(60,21);Textcolor(7);Textbackground(0);
      gotoxy(44,23);read(Tipo);
      Desicion;
    end
  else
    begin
      if tipo = '1' then Op := 0;
      if tipo = '2' then Op := 1;
    end;
end;
```

```
procedure Captura;
begin
  clrscr;
  gotoxy(25,5);writeln('MINIMOS CUADRADOS MODELO LINEAL');
  gotoxy(33,8);writeln('Tipo de modelo:');
  gotoxy(27,13);writeln('1.- Con ordenada al origen');
  gotoxy(27,16);writeln('2.- Sin ordenada al origen');
  gotoxy(35,23);write('Opción [_]');
  Desecton;
  clrscr;
  if Op = 0 then mensaje := mensaje1
            else mensaje := mensaje2;
  gotoxy(25,2);writeln('MINIMOS CUADRADOS MODELO LINEAL');
  gotoxy(26,5);writeln('Modelo ',mensaje);
  gotoxy(20,7);writeln('Número de variables independientes: nv = ');
  gotoxy(61,7);read(nv);
  if nv = 1 then
    begin
      gotoxy(17,9);
      writeln('Grado de polinomio para una sola variable: G = ');
      gotoxy(64,9);read(gp);
    end;
  gotoxy(30,11);writeln('Número de datos: n = ');
  gotoxy(51,11);read(n);
  if n <= 10 then
    for j := 1 to n do
      begin
        for i := 1 to nv do
          begin
            gotoxy(7,12+j);write('x[',i,',',j,'] = ');
            gotoxy(16,12+j);read(x[i,j]);
          end;
        gotoxy(22,12+j);write('y[',j,'] = ');
        gotoxy(29,12+j);read(y[j]);
      end
    else
      for j := 11 to n do
        begin
          for i := 1 to nv do
            begin
              gotoxy(42,12+j);write('x[',i,',',j,'] = ');
              gotoxy(50,12+j);read(x[i,j]);
            end;
          gotoxy(62,12+j);write('y[',j,'] = ');
          gotoxy(70,12+j);read(y[j]);
        end;
  gotoxy(21,24);writeln('Presione cualquier tecla para continuar');
  gotoxy(60,24); c := readkey;
  clrscr;
  gotoxy(25,1);writeln('MINIMOS CUADRADOS MODELO LINEAL');
  gotoxy(26,2);writeln('Modelo ',mensaje);
  gotoxy(20,4);writeln('Matriz ampliada del sistema de ecuaciones');
end;
```

```
procedure Proceso;
begin
      for j := n+1 downto k do
        begin
          a[k,j] := a[k,j] / a[k,k];
        end;
      for i := 1 to n do
      begin
       if i < > k then
                 begin
                   for j := n+1 downto k do
                     begin
                       a[i,j] := a[i,j] - a[k,j]*a[i,k];
                     end;
                 end;
      end;
end;

procedure Intercambio;
begin
     for j := k to n do
       if a[j,k] < > 0 then
                    begin
                      if j < > k then
                                  begin
                                    for i := k to n+1 do
                                      begin
                                        AA := a[j,i];
                                        a[j,i] := a[k,i];
                                        a[k,i] := AA;
                                      end;
                                    proceso;
                                  end
                                else
                                  proceso;
                    end;
end;
procedure GaussJordan;
begin
  for k := 1 to n do
    begin
      Intercambio;
    end;
end;

procedure MinimosCuadrados;
begin
  m := nv;
  if gp < > 0 then
      begin
        m := gp;
        for j := 1 to n do
          for i := 2 to gp do
              x[i,j] := x[i-1,j] * x[1,j];
      end;
```

```
  for k := 1-Op to m do
    begin
      for i := 1-Op to m  do
        for j := 1 to n do
          a[k + Op,i + Op] := a[k + Op,i + Op] - x[i,j]*x[k,j];
      for i := 1 to n do
        a[k + Op,m + Op + 1] := a[k + Op,m + Op + 1] - y[j]*x[k,j];
    end;
    for i := 1 to 3  do
      begin
        for j := 1 to 4 do
          begin
            gotoxy(5 + paso,5 + j);write('a',i,j,' = ',a[i,j]:6:10);
          end;
        paso := 25*i;
      end;
  ne := m + Op;
  GaussJordan;
end;

procedure Resultado;
begin
  gotoxy(25,11);writeln('Resolucion: metodo Gauss-Jordan');
  paso := 0;
  for i := 1 to 3  do
    begin
      for j := 1 to 4 do
        begin
          gotoxy(5 + paso,12 + j);write('a',i,j,' = ',a[i,j]:6:10);
        end;
      paso := 25*i;
    end;
  paso := 0;
  gotoxy(22,18);writeln('Ecuacion de regresion:   Coeficientes');
  for i := 1 to 3 do
    begin
      gotoxy(5 + paso,21);write('a',i-OP,' = ',a[i,4]:4:10);
      paso := 25*i;
    end;
  gotoxy(20,24);write('Presione  cualquier tecla para continuar');
  gotoxy(60,24);c := readkey;
end;

begin
  Inicializa;
  Captura;
  MinimosCuadrados;
  Resultado;
end.
```

POLINOMIOS ORTOGONALES

Se considera el método de mínimos cuadrados para ajustar un conjunto de datos, también existe el problema para aproximar funciones. Consideraremos el segundo caso cuando la aproximación de mínimos cuadrados es llevada a cabo sobre un intervalo [a,b]. E intentamos primero construir un conjunto de polinomios $P_1(x)$, $P_2(x)$, ... $P_n(x)$ tal que cada uno de ellos sea ortogonal a todos los demás en el conjunto sobre [a,b] relativo a la función peso especificada w(x) la cual es no negativa en el intervalo. Esto es conveniente para formular que $P_n(x)$ es un polinomio de grado n. El problema se podrá resolver, en particular, si consideramos un polinomio de grado inferior a n entonces $P_n(x)$ es un polinomio de grado n.

$$\int_a^b w(x)\, P_n(x)\, y_k(x)\, dx = 0 \qquad (1)$$

donde $k > n$, además $w(x) = > 0$ y ya que y_k es un polinomio arbitrario ortogonal a todos los demás que son representados por $P_n(x)$, puede ser escrito como una suma de y_n, y_1, ..., y_r con $r>k$. De acuerdo a la necesidad de expresarlo de una mejor forma, integramos por partes n veces, utilizando el factor $y_k^{(n)}=0$. Para este propósito presentamos la siguiente notación

$$w(x)\, P_n(x) = \frac{d^n\, U_r(x)}{dx^n} \qquad (2)$$

de este modo adquiere la forma

$$\int_a^b U_n^{(n+1)}(x)\, y_k(x)\, dx = 0$$

o después de n integraciones por partes

$$[U_n^{(n-1)} y_k - U_n^{(n-2)} y_k' + U_n^{(n-3)} y_k'' - \ldots - (-1)^{n-1} U_n y_k^{(n-1)}]_a^b = 0 \qquad (3)$$

La necesidad que la función $P_n(x)$ definida por (2)

$$P_n(x) = \frac{1}{W(x)} \frac{d^n U_n(x)}{dx^n} \qquad (4)$$

a un polinomio de grado n implicando que $U_n(x)$ debe satisfacer la ecuación diferencial

$$\frac{d^{n+1}}{dx^{n+1}} \left[\frac{1}{W(x)} \frac{d^n U_n(x)}{dx^n} \right] = 0 \qquad (5)$$

en [a,b], por cuanto el requerimiento de satisfacer a (3) por algunos valores de

$$y_k(a),\ y_k'(a),\ y_k''(a),\ \ldots,\ y_k^{(n-1)}(a)$$

y así sucesivamente, se acerca a satisfacer las 2n condiciones a la frontera.

$$\begin{aligned} U_n(a) &= U_n'(a) = U_n''(a) = \ldots = U_n^{(n-1)}(a) = 0 \\ U_n(b) &= U_n'(b) = U_n''(b) = \ldots = U_n^{(n-1)}(b) = 0 \end{aligned} \qquad (6) \text{ y } (7)$$

De este modo si, para cada entero n, y (7) podemos obtener una solución de (5) la cual satisfaga (6) y (7), el enésimo miembro del conjunto requerido de funciones esta dado por (4).

La homogeneidad de estas condiciones, sirve a fin de que cada solución pueda contener una constante multiplicativa arbitraria. Se sabe que el problema así formulado realiza

efectivamente la posesión de una solución (no terminal), igual cuando a y/o b son infinitas, bajo la suposición que $w(x) => 0$ en el intervalo y que

$$\int_a^b x^k w(x)\,dx$$

existe para todos los valores de k no negativos de la integral.

Los coeficientes en la expresión

$$y(x) = \sum_{n=0} a_n P_n(x) \qquad (8)$$

son determinados por el requerimiento

$$\int_a^b w(x)\,[f(x) - y(x)]^2 dx = MIN \qquad (9)$$

en la forma

$$a_n = \frac{\int_a^b wfP_n dx}{\int_a^b wP_n^2 dx} = \frac{\int_a^b wfP_n dx}{\gamma_n} \qquad (10)$$

donde

$$\gamma_n = \int_a^b wP_n^2 dx \qquad (11)$$

a pesar de que el numerador en (10) depende de f, el denominador γ_n es independiente de f y puede ser determinado de una vez, y para todos.

El calculo de γ_n se facilita con las siguientes consideraciones. Si escribimos

$$P_n(x) = \Delta_0 + \Delta_1 x + \ldots + \Delta_n x^n \qquad (12)$$

de modo tal que Δ_{in} es el coeficiente de x^i en $P_n(x)$ y $\Delta_{ii}=\Delta_{in}$ sea el coeficiente dominante, en cuanto a eso continuamos

$$\gamma_n = \int_a^b w(x) P_n(x) P_n(x)\, dx$$

$$= \int_a^b w(x) P_n(x) [\Delta_0 + \Delta_1 + \ldots + \Delta_n x^n]\, dx$$

y por lo tanto si recordamos las relaciones

$$\int_a^b w(x) P_n(x) x^i dx = 0, \qquad i = 0, 1, \ldots n-1 \qquad (13)$$

lo cual es equivalente a (1), podemos deducir que

$$\gamma_n = \Delta_n \int_a^b x^n w(x) P_n(x)\, dx = \Delta_n \int_a^b x^n U_n^{(n)}(x)\, dx$$

Mediante integración por partes n veces y haciendo uso de (6) y (7), esta relación tiene la forma conveniente

$$\gamma_n = \int_a^b w(x) P_n^2(x)\, dx = (-1)^n n! \Delta_n \int_a^b U_n(x)\, dx \qquad (14)$$

donde Δ_n es el coeficiente de x^n en $P_n(x)$.

Se advierte que el problema especificado por (8) y (9) puede ser generalizado de la siguiente manera. Puede ser que f(x) sencillamente no pueda ser aproximado satisfactoriamente sobre [a,b] por un polinomio de bajo grado, pero que pueda encontrarse una cierta función, v(x), conocida tal que la relación f(x)/v(x) pueda ser de tal manera aproximada. De este modo, si determinamos los coeficientes del polinomio

$$y(x) = \sum_{n=1} b_n P_n(x) \tag{15}$$

de modo que

$$\int w(x) \left[\frac{f(x)}{v(x)} - \sum_{n=1} b_n P_n(x) \right]^2 dx = MIN \tag{16}$$

la ortogonalidad de los polinomios relativa a w nos conduce al resultado

$$b_n = \frac{1}{\gamma_n} \int \frac{w}{v} f P_n dx \tag{17}$$

Esto es, ver que (16) es equivalente al resultado de minimizar el cuadrado del error (f-vy)²; con la función ponderando w/v². Frecuentemente se selecciona w(x) = v(x) con buenos resultados.

Algoritmo:

Para determinar los coeficientes $a_0, \ldots, a_m$ del polinomio de mínimos cuadrados de grado m, dado un conjunto de $n+1$ valores $(x_0, f(x_0)), \ldots, (x_n, f(x_n))$:

Entrada: número de pares de datos n;

datos x_i e y_i.

Salida: Los coeficientes del polinomio.

Paso 1 Para i = 1, ..., n realizar

Tomar $P_{0,i} = 1$

Tomar acum = acum + x_i

Paso 2 Tomar acum = acum/n

Paso 3 Tomar γ_1 = acum

Paso 4 Para i = 1, ..., n hacer

Tomar $P_{1,i} = x_i - \text{acum}$

Paso 5 Para k = 2, ..., m realizar

Tomar acum = 0; sl = 0

Para i = 1, ..., n hacer

acum = acum + $x_i * P^2_{k-1,i}$

sl = sl + $P^2_{k-1,i}$

Tomar γ_k = suma/sl; suma = 0; sl = 0

Para i = 1, ...n realizar

Tomar acum = acum + $x_i * P_{k-1,i} * P_{k-2,i}$

Tomar sl = sl + $P^2_{k-1,i}$

Tomar δ_k = acum/sl;

Para i, ... n hacer

Tomar $P_{k,i} = (x_i - \gamma_k)P_{k-1,i} - \delta_k * P_{k-2,i}$

Paso 6 Para $k = 0, \ldots, m$ realizar

Tomar acum = 0; sl = 0

Para $i = 1, \ldots, n$ hacer

Tomar acum = acum + $P_{k,i} * y_i$

Tomar sl = sl + $P^2_{k,i}$

Tomar D_k = acum/n

Paso 7 Tomar acum = 0

Paso 8 Para $i = 1, \ldots, n$ realizar

Tomar sl = 0

Para $j = 0, \ldots, m$ hacer

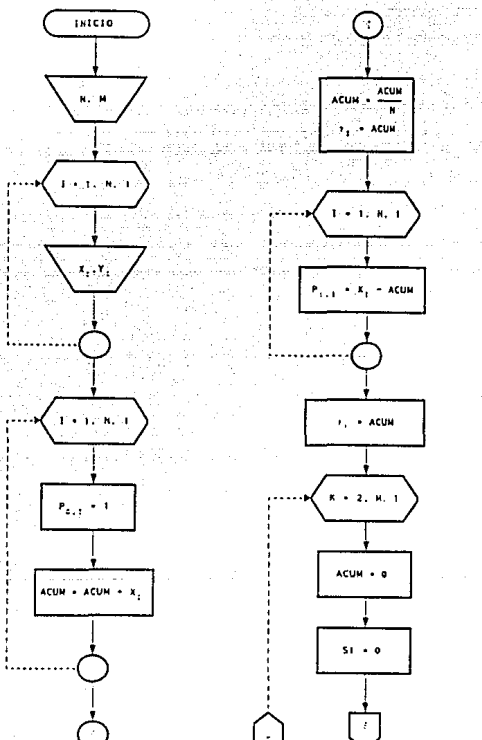
sl = sl + $D_j + P_{j,i}$

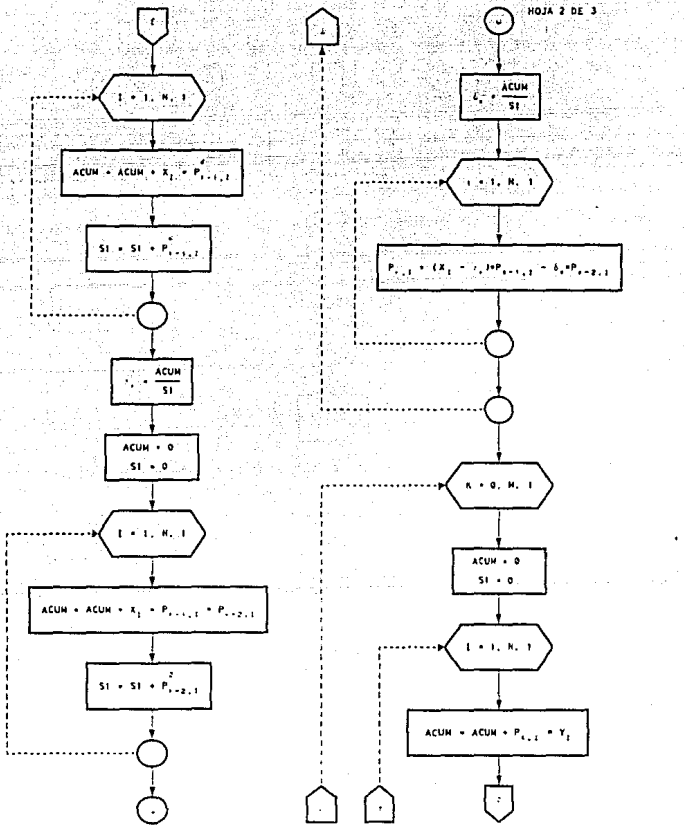
Tomar acum = acum + $(y_i - sl)^2$

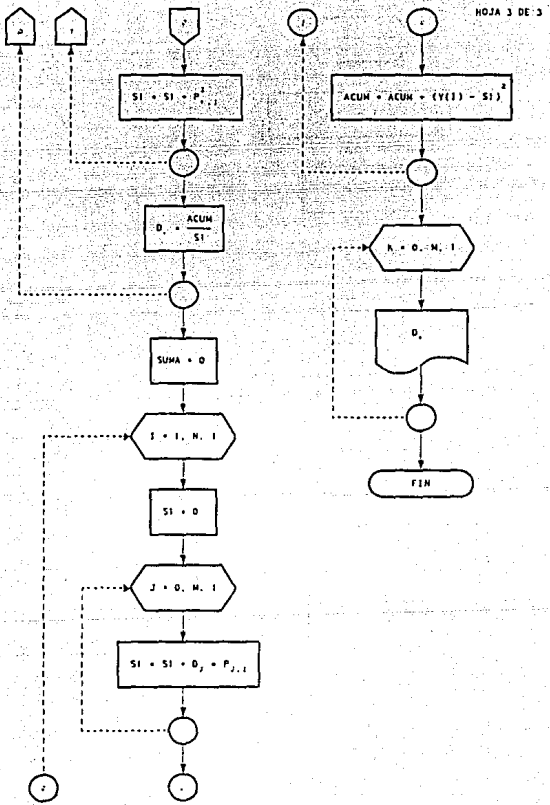
Paso 9 Salida (Para $k = 0, \ldots, m$; Imprimir D_k)

Terminar.

POLINOMIOS ORTOGONALES




C
I = 1, N, 1
ACUM = ACUM + X_I * P_{I-1,I}^K
SI = SI + P_{I-1,I}^K
C_K = ACUM / SI
ACUM = 0
SI = 0
I = 1, N, 1
ACUM = ACUM + X_I * P_{I-1,I}^Z * P_{I-2,I}
SI = SI + P_{I-2,I}^Z
A
B
D
C_K = ACUM / SI
I = 1, N, 1
P_{I-1,I} = (X_I - r_I)*P_{K-1,I} - δ_K*P_{K-2,I}
K = 0, N, 1
ACUM = 0
SI = 0
I = 1, N, 1
ACUM = ACUM + P_{I-1,I} * Y_I
E


SI ← SI + P²J,I
ACUM ← ACUM + (Y(I) − SI)²
Dᵢ ← ACUM / SI
K ← 0, N, 1
Dᵢ
SUMA ← 0
I ← 1, N, 1
SI ← 0
J ← 0, M, 1
SI ← SI + Dⱼ * PJ,I
FIN

```
Program Minimos_Cuadrados_Polinomios_Ortogonales;
uses Crt;
var i, j, k, m, n : Integer;
    suma, S1  : real;
    Gamma, Delta, D, x, y : array [0..20] of real;
    P : array [0..10,1..20] of real;
    c: char;

procedure Inicializa;
begin
  i := 0;
  j := 0;
  n := 0;
  k := 0;
  suma := 0;
  S1 := 0;
  c := char(0);
  for i := 1 to 20 do
    begin
      Gamma[i] := 0;
      Delta[i] := 0;
      D[i]     := 0;
      x[i]     := 0;
      y[i]     := 0;
    end;
  for i := 0 to 10 do
    for j := 1 to 20 do
      P[i,j] := 0;
end;

procedure  Captura;
begin
  clrscr;
  gotoxy(32,1);write('MINIMOS CUADRADOS');
  gotoxy(30,2);write('POLINOMIOS ORTOGONALES');
  gotoxy(25,4);write('Suministre los siguientes datos');
  gotoxy(28,6);write('Grado de polinomio: m = ');
  gotoxy(52,6);read(m);
  gotoxy(29,8);write('Número de datos: n = ');
  gotoxy(50,8);read(n);
  for i := 1 to n do
    begin
      gotoxy(20,9 + i);write('x[',i,'] = ');
      gotoxy(51,9 + i);write('y[',i,'] = ');
    end;
  gotoxy(21,24);
  write('Para continuar presione cualquier tecla');
  for i := 1 to n do
    begin
      gotoxy(27,9 + i);read(x[i]);
```

```
      gotoxy(58,9+i);readln(y[i]);
    end;
  gotoxy(60,24);c := readkey;
end;

procedure PolOrto;
begin
  clrscr;
  for i := 1 to n do
    begin
      suma := suma + x[i];
      P[0,i] := 1;
    end;
  suma := suma/n;
  Gamma[1] := suma;
  for i := 1 to n do
    P[1,i] := x[i] - suma;
  for k := 2 to m do
    begin
      suma := 0;
      S1   := 0;
      for i := 1 to n do
        begin
          suma := suma + x[i]*Sqr(P[k-1,i]);
          S1   := S1 + Sqr(P[k-1,i]);
        end;
      Gamma[k] := suma/S1;
      suma := 0;
      S1   := 0;
      for i := 1 to n do
        begin
          suma := suma + x[i]*P[k-1,i]*P[k-2,i];
          S1 := S1 + Sqr(P[k-2,i]);
        end;
      Delta[k] := suma/S1;
      for i := 1 to n do
        P[k,i] := (x[i]-Gamma[k])*P[k-1,i]
                  - Delta[k]*P[k-2,i];
    end;
  for k := 0 to m do
    begin
      suma := 0;
      S1   := 0;
      for i := 1 to n do
        begin
          suma := suma + P[k,i]*y[i];
          S1 := S1 + Sqr(P[k,i]);
        end;
      D[k] := suma/S1;
    end;
```

```
  suma := 0;
  for i := 1 to n do
    begin
      S1 := 0;
      for j := 0 to m do
        S1 := S1 + D[j]*P[j,i];
      suma := suma + Sqr(y[i]-S1);
    end;
end;

procedure Resultado;
begin
  clrscr;
  gotoxy(32,2);write('MINIMOS CUADRADOS');
  gotoxy(30,4);write('POLINOMIOS ORTOGONALES');
  gotoxy(25,7);write('Los datos suministrados fueron:');
  for i := 1 to n do
    begin
      gotoxy(20,9+i);write('x[',i,'] = ',x[i]:4:4);
      gotoxy(53,9+i);write('y[',i,'] = ',y[i]:4:4);
    end;
  for k := 0 to m do
    begin
      gotoxy(25,17+k);write('Coeficiente a[',k,'] = ',D[k]:6:10);
    end;
  gotoxy(27,21);write('Suma residual: ',suma:6:10);
  gotoxy(21,24);write('Para continuar presione cualquier tecla');
  gotoxy(60,24);c := readkey;
end;

begin
  Inicializa;
  Captura;
  PoliOrto;
  Resultado;
end.
```

CAPITULO VIII

CONCLUSIONES

CONCLUSIONES

La presente tesis inicia con una introducción breve sobre varios temas que son fundamentales para una mejor comprensión de los métodos numéricos.

Para conformar este trabajo se consideró la necesidad de implementar técnicas numéricas que nos ayudarán a resolver los problemas que se presentan tanto durante la carrera, como en el ejercicio profesional de la Ingeniería Química, de acuerdo a esto se seleccionaron las técnicas más útiles para nuestros requerimientos.

En el proceso de programación y codificación es de suma importancia seguir una metodología, partiendo de una base teórica que nos dé soporte para la creación de un algoritmo ó secuencia lógica para realizar el proceso y su representación en lenguaje simbólico (diagrama de flujo), los cuales nos permiten codificar el programa en cualquier lenguaje de programación, dando a estos elementos mayor relevancia por su dinamismo.

CRITERIOS GENERALES DE COMPARACIÓN

Con la finalidad de obtener la mejor técnica numérica de cada tema he tomado en cuenta los siguientes criterios:

Convergencia: Seguridad y eficiencia para encontrar la raíz.

Rapidez: El menor número de iteraciones, reduciendo el tiempo de proceso.

Tolerancia: Asignar el valor adecuado tal que la aproximación a la raíz esté dentro de la precisión deseada.

Función: Que sea reducible a cero y pueda reescribirse mediante manipulaciones algebraicas.

Estabilidad: La no acumulación y propagación del error dentro del proceso.

La convergencia y rapidez se ven afectadas por varios aspectos:

La tolerancia, generalmente cuando ésta es muy pequeña se refleja en un gran número de iteraciones para encontrar la aproximación a la solución, lo cual nos dará un mayor tiempo de proceso. Por otro lado una tolerancia grande podría hacernos pasar cerca de la raíz y seguir de frente, es decir que el método diverja.

La lentitud propia del método en cuanto a operaciones que se tengan que realizar, aunque

sí converja a la solución,

Que el método sea deficiente.

La codificación incorrecta del método.

La complejidad algebraica de la función.

Considerando lo anterior analizo cada tema para finalizar sugiriendo una o más técnicas que juzgo las más apropiadas para el tratamiento de problemas relacionados con esa área.

En el caso de interpolación las técnicas presentadas se enfocan a resolver diferentes problemas de la misma área siendo estas:

Interpolación de Lagrange.

Interpolación en dos variables.

Interpolación cúbica por segmentos.

Para el tema de resolución de ecuaciones algebraicas no lineales se toman en cuenta los aspectos anteriormente expuestos. De este modo sugiero el método de Newton con la salvedad de que la derivada de la función no sea compleja.

En cuanto a integración recomiendo el método de Romberg, aunque requiere un poco más de tiempo, pero es más exacto.

Para resolver ecuaciones diferenciales ordinarias aconsejo el empleo de los siguientes métodos:

Runge - Kutta de 4° orden y 7 constantes;

Runge - Kutta de 4° orden y 10 parámetros;

Milne de 6° orden.

En sistemas de ecuaciones algebraicas lineales se hace necesaria la distinción entre matrices densas o cubiertas (de orden menor a 30) y dispersas (orden mayor a 30); para las primeras el método de Gauss - Jordan es el sugerido y para las segundas el método de Gauss - Seidel aunque este último no siempre converge.

Por otro lado en sistemas de ecuaciones algebraicas no lineales recomiendo el método de Newton-Raphson haciendo la misma advertencia que para una ecuación.

Y para sistemas de ecuaciones diferenciales ordinarias el método de Runge - Kutta de 4° orden y 7 constantes.

En el capítulo de ajuste de datos se exponen técnicas para obtener los coeficientes de una ecuación que aproxima a una función representada por datos, en este tema aconsejo el método del polinomio de interpolación y el método de mínimos cuadrados para modelos de polinomios ortogonales.

Es importante notar que también se pueden obtener mejores resultados al implementar dos o más métodos, como en algunas de las técnicas presentadas (Método de Romberg, Métodos multipaso, Métodos predictores-correctores).

La codificación la desarrollo en base a las cualidades del lenguaje Pascal (Turbo Pascal): Transparencia, pues permite escribir programas totalmente claros en su propósito; Seguridad, por la facilidad en la detección de errores; y Eficiencia, por la estructura de su diseño.

Finalmente en la selección de el método para la resolución de problemas a través de microprocesadores, se tiene que tomar en cuenta, además de lo anterior, las características del problema y el tiempo de proceso pues esta técnica puede ser usada varias veces por el programa principal incrementado el tiempo total de ejecución.

Así pues, creo que este trabajo resulta de gran valía tanto para los estudiantes como para los profesionales de la Ingeniería Química.

BIBLIOGRAFIA

BIBLIOGRAFIA

1. *Numerical Methods for Partial Differential Equations*
   Ames, William F.
   2ª Ed.
   Academic Press Inc.
   USA 1977.

2. *An Introduction to Numerical Analysis*
   Atkinson, Kendall E.
   2ª Ed.
   John Wiley & Sons.
   Singapur 1989.

3. *Introducción a los Métodos Numéricos con Pascal*
   Atkinson L. V.
   Harley P.J.
   Addison Wesley Iberoamericana.
   México 1987.

4. *Fundamentos de Matemáticas Superiores*
   Ayres, Frank Jr.
   Ed. McGraw-Hill (Serie Schaums).
   Colombia 1969.

5. *Numerical Methods*
   Bakhalov N.S.
   1ª Ed.
   MIR Moscú.
   URSS 1973

6. *Computer Graphics with Pascal*
Berger, Marc.
The Benjamin/Cummings Publishing Company Inc.
USA 1990.

7. *Pascal Precisely for Engineers and Scientists*
Bishop Judy
Bishop Miguel.
Addison-Wesley Publishing Co.
Gran Bretaña 1990.

8. *Introducción a las Ecuaciones Diferenciales*
Boyce, William E.
Di Prima, Richard C.
6ª reimp.
LIMUSA
México 1986.

9. *Basic Matrices*
Brayden C.G.
MacMillan Press LTD.
Inglaterra 1975.

10. *Análisis Numérico*
Burden Richard L.
Faires J. Douglas.
Grupo Editorial Iberoamericana.
México 1986.

11. *An Introduction to Differencial Equations*
Campbell Stephen.
Wadsworth Publishing.
2ª Ed.
USA 1990.

12. *Cálculo Numérico, Métodos y Aplicaciones*
Carnahan Brice,
H.A. Luther,
James O. Wilkes,
Editorial Rueda.
España.

13. *Numerical Solution of Initial Value Problems*
Ceschino F.
Kuntzmann J.
Prentice Hall, Inc.
USA 1968.

14. *Numerical Mathematic and Computing*
Chiney E. Ward
Kincaid David.
Book/Cole Publishing Co.
USA.

15. *Funtional Analysis and Numerical Mathematics*
Collatz Lothar.
Academic Press, Inc.
USA 1966.

16. *Métodos y Algoritmos Básicos del Algebra Numérica*
Conde Lázaro Carlos
Winter Althaus.
Reverte S.A.
España 1990.

17. *Elementary Numerical Analysis*
Conte Samuel D.
De Boor Carl.
3ª Ed.
McGraw-Hill Book Co.
Singapur 1980.

18. *Análisis Numéricos*
Curtis F. Gerald.
Representaciones y Servicios de Ingeniería, S.A.
2ª Ed.
México 1987.

19. *Ecuaciones Diferenciales con Aplicaciones*
Derrick William R.
Crossman Stanley J.
Addison-Wesley Iberoamericana.

20. *Métodos Numéricos*
Di Constanzo Lorenzez Ruba Elena.
Scheid Franck.
2ª Ed.
McGraw-Hill.
México 1991.

21. *Ordinary Differential Equations with Modern Applications*
Finizio N.
Ladas G.
3ª Ed.
Wadsworth Publishing Company.
USA 1979.

22. *Computer Solution of Linear Algebraic Systems*
Forsythe George E.
Moler Cleve B.
Prentice Hall, Inc.
USA 1967.

23. *Modeling and Simulation in Chemical Engineering*
Franks, Roger G. E.
1ª Ed.
Wiley-Interscience.
USA 1972.

24. *Systems of Ordinary Differential Equations an Introduction*
Goldberg, Jack L.
Schwarte, Arthur J.
Harper & Row Publishers.
USA 1972.

25. *Numerical Analysis for Applied Mathematics, Science and Engineering*
Greenspan Donald.
Casulli Vicenzo.
Addison-Wesley Publishing Co.
USA 1988.

26. *Introduction to Applied Numerical Analysis*
Hamming, Richard Wesley.
International Student Edition.
McGraw-Hill Kogakusha, LTD.
Japón 1971.

27. *Numerical Methods For Scientist and Engineers*
Hamming, Richard Wesley.
2ª Ed.
Dover Publication, Inc.
USA 1987.

28. *Elementos de Análisis Numérico*
Henrici Peter.
1ª reimp.
Trillas.
México 1977.

29. *Introduction to Numerical Analysis*
Hildebrand F.B.
2ª Ed.
Tata McGraw-Hill Publishing.
India 1974.

30. *Apuntes de Métodos Numéricos*
Iriarte B. Rafael
Borras G. Hugo E.
Duran C. Rossynela.
Facultad de Ingeniería, UNAM.
México 1983.

31. *Numerical Analysis*
Jacques Jan.
Judd Colin.
Chapman and Hall.
Gran Bretaña 1987.

32. *Numerical Solutions of Differetial Equations*
Jain, M. K.
John Wiley & Sons.
India 1979.

33. *Métodos Numéricos Aplicados a la Computación Digital con Fortran*
James M.L. et al
Representaciones y Servicios de Ingeniería.
México 1970.

34. *Métodos Matemáticos en Ingeniería Química*
Jenson V.O.
Jefries G.V.
1ª Ed.
Alhambra S.A.
España 1969.

35. *Matrices, Aplicaciones Matemáticas en Economía y Administración*
Kleiman Ariel
Kleiman Elena K.
LIMUSA. México 1990.

36. *Ecuaciones Diferenciales*
Kreider, Donald L.
Kuller, Robert G.
Ostberg, Donald R.
Fondo Educativo Interamericano.
México 1973.

37. *Numerical Analysis*
Kunz Kaiser.
McGraw-Hill Book Co.
USA 1957.

38. *Computer Applications of Numerical Methods*
Kuo Shans
Addison-Wesley Publishing Co.
USA 1972.

39. *Computational Methods in Ordinary Differential Equations*
Lambert J.D.
John Wiley & Sons, Inc.
Gran Bretaña 1973.

40. *Turbo Pascal Programing and Problem Solving*
Leestma, Sanford.
Nyhoff, Larry.
3ª Ed.
McMillan Publishing Company.
USA 1990.

41. *A Course in Numerical Analysis*
Lieberstein H. Melvin.
Harper & Row, Publishers, Inc.
USA 1968.

42. *The Numerical Analysis Problem Solver*
Staff of Research and Education Association.
Dr. M. Fogiel, Director.
Research and Education Association.
USA 1986.

43. *Introduction to Numerical Methods and Fortran Programming*
McCalla Thomas Richard.
John Wiley & Sons, Inc.
USA 1967.

44. *Métodos Numéricos y Programación Fortran*
McCraken Daniel D.
Dorn William S.
6ª Reimp.
LIMUSA-Wiley, S.A.
México 1973.

45. *Programación y Cálculo Numérico*
Michavila Francisco
Gavete L.
Reverte S.A.
España 1985.

46. *Numerical Solutions of Differential Equations*
Milne, William Edmund.
John Wiley & Sons.
USA 1960.

47. *The Finite Difference in Partial Differential Equations*
Mitchell, A. R.
Griffits, D. F.
John Wiley & Sons.
Inglaterra 1980.

48. *Elementary Theory and Application of Numerical Analysis*
Moursund David G.
Duris Charles S.
Dover Publication Inc.
USA 1988.

49. *Métodos Numéricos*
Olivera Salazar Antonio.
Coordinación de Servicios Académicos.
Facultad de Ingeniería, UNAM.
México 1972.

50. *A First Course in Numerical Analysis*
Ralston Anthony.
McGraw-Hill Book Co.
USA 1965.

51. *Introducción al Análisis Numérico*
Ralston Anthony.
LIMUSA-Wiley, S.A.
1ª Ed.
México 1970.

52. *Numerical Methods, Software and Analysis*
Rice, John R.
McGraw-Hill Book Co.
Singapur 1985.

53. *Differential Equations*
Ross, Shepley L.
3ª Ed.
John Wiley & Sons, Inc.
USA 1984.

54. *Análisis Numérico*
Scheid Francis.
McGraw-Hill.
Colombia 1972.

55. *Numerical Computing an Introduction*
Shampine Lawrence R.
Allen Jr. Richard C.
W.B. Sanders Co.
USA 1973.

56. *Mathematics of Physics an Modern Engineering*
Sokolnikoff, I. S.
Redheffer, R. M.
McGraw-Hill Book Company.
USA 1958.

57. *Introduction to Numerical Analysis*
Weeg Gerard P.
Reed Georgia B.
Blaisdell Publishing Co.
USA 1966.

58. *Numerical Computation*
Williams P.W.
Nelson.
Gran Bretaña 1972.

59. *A Surved of Numerical Mathematics*
Young David M.
Gregory Robert Todd.
Addison-Wesley Publishing Co.
Vol I,II.
USA 1972.

60. *Ecuaciones Diferenciales con Aplicaciones*
Zill Dennis A.
Grupo Editorial Iberoamericana.
México 1988.

INDICE

INDICE