11
2 ejm

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

"SOLUCION DE SISTEMAS DE ECUACIONES LINEALES DE VANDERMONDE"

T E S I S

Que para obtener el título de

M A T E M A T I C O

p r e s e n t a

CARLOS GOMEZ ALONSO

Director de Tesis: DOCTOR CARLOS J.E. SIGNORET POILLON

México, D. F. 1990

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

VNIVERSIDAD NACIONAL
AVPNOMA DE
MEXICO

M. en C. Virginia Abrín Batule
Jefe de la División de Estudios Profesionales de la
Facultad de Ciencias
P r e s e n t e

Comunicamos a usted que hemos revisado el trabajo de Tesis:

"SOLUCION DE SISTEMAS DE ECUACIONES LINEALES DE VANDERMONDE"

realizado por CARLOS GOMEZ ALONSO

con número de cuenta 7962043-7 , pasante de la carrera de MATEMATICO

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis Propietario DOCTOR CARLOS J.E. SIGNORET POILLON

Propietario M. en C. JOSE WILLIAM GALLARDO

Propietario M. en C. ALEJANDRO BRAVO MOJICA

Suplente M. en C. EDGAR RENE HERNANDEZ MARTINEZ

Suplente ACT. MA. del ROSARIO ESPINOZA TUFIÑO

Consejo Departamental de Matemáticas

DOCTOR MANUEL FALCONI MAGAÑA

CONTENIDO

0. Introducción

La resolución de sistemas de ecuaciones lineales es un problema clásico en el álgebra. Aunque desde el punto de vista teórico este problema ha sido totalmente resuelto, desde el punto de vista operativo continúa siendo objeto de estudio y fuente de ideas y algoritmos. Diversos métodos se han propuesto para resolver sistemas de ecuaciones lineales que tienen algun tipo particular; dentro de estos tipos especiales se encuentran los *Sistemas de Vandermonde* (por Vincent Téophile Vandermonde, matemático francés del siglo XVI).

Una matriz V de la forma

$$V = V(\alpha_0,\ldots,\alpha_n) = \begin{bmatrix} 1 & 1 & . & 1 \\ \alpha_0 & \alpha_1 & . & \alpha_n \\ \alpha_0^2 & \alpha_1^2 & . & \alpha_n^2 \\ . & . & & . \\ \alpha_0^n & \alpha_1^n & . & \alpha_n^n \end{bmatrix}, \alpha_0,\alpha_1,\ldots,\alpha_n \in \mathbb{R}$$

es llamada una *matriz de Vandermonde* y un sistema de ecuaciones lineales de la forma V x = b (sistema *primal*) ó V^T a = f (sistema *dual*) es llamado un *sistema de ecuaciones lineales de Vandermonde.*

Por el tipo especial de la matriz de Vandermonde, la resolución de estos sistemas es particularmente cómoda en el ambiente de los métodos numéricos, principalmente mediante el uso de la interpolación de Newton.

En el año de 1967 un trabajo fué desarrollado y publicado por C. Ballester & V. Pereyra sobre construcción de aproximaciones discretas [1] a partir del cual fué deducido un método con el objeto de encontrar solución al sistema de ecuaciones lineales con coeficientes de tipo polinomial. Dicho método fué posteriormente implementado y mejorado [2], requiriendo menos memoria y ejecutando el proceso en menor tiempo. También Galimberti & Pereyra [6] utilizaron este mismo método para la resolución de sistemas de Vandermonde.

Si en la matriz de Vandermonde V aparecen algunos argumentos α_i repetidos, entonces decimos que estamos en el *caso confluente*. Si se ha resuelto un sistema de Vandermonde de tamaño $n \times n$ y se agrega un nuevo

argumento, es decir , se agrega a la matriz del sistema la última columna y el último renglón (obteniendo así una matriz $(n+1)\times(n+1)$) decimos que estamos en el *caso progresivo.*

Estudios publicados en los años siguientes ([2],[3] y [4]) muestran los algoritmos para los distintos casos de sistemas de Vandermonde. Aunque estos procesos son esencialmente iterativos, tienen un gran contenido algebraico, en especial el algoritmo que se usa en el caso *primal* donde la matriz original es reducida a la forma de bloques triangulares (PA=LU) con bloques diagonales.

En los primeros apartados (I,II,III,IV) del presente trabajo, presentamos los prerrequisitos, es decir, el material concerniente a interpolación de Newton, diferencias divididas, descomposición PA=LU de una matriz, y el esquema de Horner.

Luego en el capítulo 1 tratamos los sistemas no-confluentes en sus dos casos, dual y primal, así como también los algoritmos progresivos. En el capítulo 2 tratamos con sistemas confluentes, también en sus dos casos, primal y dual.

En cada capítulo hemos optado por presentar los algoritmos mediante un proceso general y luego un desarrollo de procedimientos. En el capítulo 3 presentamos ejemplos particulares numéricos. También presentamos un breve análisis de la eficiencia de los algorimos desarrollados; Apéndice A.

Finalmente, presentamos la implementación de los algoritmos de este trabajo, éstos se codificaron en lenguaje de programación FORTRAN IV y se ejecutaron en una computadora del tipo PC-386 con tarjeta de coprocesador matemático 80387.

I. Método de Newton para Interpolación Polinomial

Supongamos que tenemos una tabla con $n+1$ valores numéricos $(x_0, y_0),(x_1, y_1),\ldots,(x_n, y_n)$ de la gráfica de una función real dada

$$y = f(x)$$

representados así, donde los x_i son diferentes entre sí :

x	x_0	x_1	$\cdots$	x_n
y	y_0	y_1	$\cdots$	y_n

Podemos aproximar $f(x)$ con una función polinomial $p(x)$. $p(x)$ es usado para aproximar $f(x)$. Entonces podemos determinar un polinomio $P_n(x_i) = y_i$ de grado máximo $\leq n$ que involucre las funciones de valor y_i para calcular los coeficientes en la *Interpolación polinomial* determinados por

$$P_n(x_i) = y_i$$

El problema de Interpolación es encontrar una curva polinomial $y = p(x)$ que pase por los puntos de coordenadas (x_i, y_i), $0 \leq i \leq n$.

Estas parejas de puntos se encuentran localizados en el plano cartesiano. Se trata de esbozar una curva que satisfaga todos los puntos, o más bien que contenga los $n+1$ puntos. Esto equivale a determinar un polinomio tal que para todos los puntos x_i , $i = 0,\ldots,n$ tome su correspondiente valor y_i para cada uno de los $n+1$ diferentes valores de x_i.

El polinomio puede variar según el numero de puntos. i.e. para $n = 1$, podemos tener un polinomio de grado cero, puesto que esto es un sólo punto $p(x_0) = y_0$; para $n = 2$, o sea dos puntos, tenemos una función lineal, la recta que pasa por dos puntos dados y que forman el polinomio :

$$p(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0)$$

En general para n puntos podemos tener un polinomio de grado $n-1$.

Supongamos que tenemos los $n+1$ puntos de los que hablamos en el plano cartesiano $p_0, p_1,\ldots, p_n$.

A lo que queremos llegar es a encontrar un polinomio $p(x)$ tal que la curva $y = p(x)$ contenga todos los puntos. Una función polinomial que es igual a la función de aproximación en un número especificado de puntos, es llamada *interpolación polinomial.*

Una forma de plantear dicho polinomio es iterativamente :

$$p(x)+c(x-x_0)(x-x_1)...(x-x_k) \qquad (*)$$

es un polinomio que pasa por los primeros k-puntos, entonces necesitamos ajustar el parámetro c, $c \in \mathbb{R}$, con la finalidad de incluir el nuevo valor y_{k+1} en x_{k+1}. Entonces :

$$p(x_{k+1})+c(x_{k+1}-x_0)(x_{k+1}-x_1)...(x_{k+1}-x_k)=y_{k+1}$$

El valor c puede irse obteniendo con este razonamiento inductivo, ya que $x_{k+1}-x_i$ para $0 \le i \le k$, no puede ser cero pues todos los x_i's son distintos. Aquí se observan dos situaciones:

Primero, que el polinomio empieza en grado cero y que cuando incrementamos un paso en este razonamiento inductivo, se incrementa en "uno" el grado del polinomio.

Segundo, que hay unicidad del polinomio p si este es de grado mínimo

Demostraremos a continuación dichas observaciones ; supongamos que existe otro polinomio q diferente a p (q es de grado a lo más $n-1$) y además satisface: $q(x_i)=y_i$ para $1 \le i \le n$; como la diferencia de polinomios es también un polinomio, $p-q$ es un polinomio de grado a lo más $n-1$ que toma el valor de cero en los $x_0,...,x_n$.

Pero un polinomio diferente de cero y con grado $n-1$, puede tener a lo más $n-1$ raíces. La conclusión es que $p=q$, lo cual establece la unicidad de p. Por lo tanto el polinomio de grado $n-1$ que pasa por todos los n diferentes puntos en el plano cartesiano, es único.

La construcción de la interpolación polinomial anterior se conoce como el *algoritmo de Newton para interpolación.*

Lo que observamos en el procedimiento al ir calculando cada una de las nuevas c's, es que el nuevo polinomio es obtenido según el algoritmo, a partir de su predecesor agregándole un nuevo término. También lo que se ve es que cuando un nuevo término se agrega, se repiten todos los anteriores:

$$p(x) = p_0(x) + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + c_3(x - x_0)(x - x_1)(x - x_2).$$

Tratando de hacerlo eficiente; $(x - x_0)$ se repite en todos los términos, excepto en el primero; $(x - x_1)$ se repite a partir del siguiente término y para todos los subsiguientes, veámoslo así más claro

$$p(x) = c_0 + c_1[\ (x - x_0)\ ] + c_2[\ (x - x_0)(x - x_1)\ ] + c_3[\ (x - x_0)(x - x_1)(x - x_2)\ ] + \ldots$$
$$\ldots + c_n[(x - x_0)(x - x_1)(x - x_2)\ldots(x - x_{n-1})],$$

ésto, en una fórmula compacta queda:

$$p(x) = c_0 + \sum_{i=1}^{n} c_i \left[\prod_{j=0}^{i-1} (x - x_j) \right]$$

Regresando a lo observado en el párrafo anterior, en donde se repiten los términos, factorizando los términos iguales queda:

$$p(x) = c_0 + (x - x_0)\Big[c_1 + (x - x_1)\Big[c_2 + (x - x_2)\Big[c_3 + (x - x_3)\Big[c_4 + \ldots (x - x_{n-1})[c_n]\Big]\Big]\Big]\Big]$$

lo que es equivalente a

$$p(x) = \Big[\ldots\Big[\Big[c_n(x - x_{n-1}) + c_{n-1}\Big](x - x_{n-2}) + c_{n-2}\Big]\ldots\Big](x - x_0) + c_0$$

Si queremos evaluar $p(t)$ en un punto t , $t \in \mathbb{R}$,empezamos del corchete más interno, al más externo hasta acabar. Ilustrando esto:

$$
\begin{aligned}
v_1 &= c_n \\
v_2 &= v_1(t - x_{n-1}) + c_{n-1} \\
v_3 &= \underbrace{[\, v_1(t - x_{n-1}) + c_{n-1} \,]}_{v_2}(t - x_{n-2}) + c_{n-2} \\
v_4 &= \underbrace{[\, v_1(t - x_{n-1}) + c_{n-1}[(t - x_{n-2}) + c_{n-2}] \,]}_{v_3}(t - x_{n-3}) + c_{n-3} \\
&\vdots \\
v_n &= v_{n-1}(t - x_0) + c_0
\end{aligned}
$$

Entonces la cantidad v_n es ahora $p(t)$. Si tenemos los puntos $x_0, x_1, \ldots, x_n$, y asumimos que son diferentes, entonces establecemos que para cada n=1,2,3..., existe un único polinomio p_{n-1} de grado $\leq n-1$ que interpola éstos puntos.

Entonces un nuevo polinomio se puede expresar en términos del polinomio anterior más un nuevo término. Así:

$$p_{n-1}(x) = p_{n-2}(x) + c_n(x - x_0)\ldots(x - x_{n-1})$$

de aquí observamos que los polinomios son inducidos a partir del anterior al ir aproximando la función $y = f(x)$ sobre el intervalo (x_0, x_n) y que satisface $p_n(x_i) = y_i$, $i = 0, 1, \ldots, n$.

$$
\begin{aligned}
f(x_0) &= c_0 \\
f(x_1) &= c_0 + c_1(x_1 - x_0) \\
f(x_2) &= c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) \\
&\vdots \\
f(x_n) &= c_0 + c_1(x_n - x_0) + c_2(x_n - x_0)(x_n - x_1) + \ldots + c_n(x_n - x_0)\ldots(x_n - x_{n-1})
\end{aligned}
$$

Aquí c_0 depende de $f(x_0)$

c_1 depende de $f(x_0)$ y $f(x_1)$

c_2 depende de $f(x_0)$, $f(x_1)$ y $f(x_2)$

$\vdots$

c_n depende de $f(x_0), f(x_1), f(x_2) \ldots f(x_n)$

A final de cuentas c_n depende de f en los puntos $x_0, x_1, \ldots, x_n$. Una notación alterna es $c_n = f[x_0, x_1, \ldots, x_n]$

Una forma de calcular explícitamente las constantes c_i se da en el próximo apartado.

II. Diferencias Divididas

Las diferencias divididas son expresiones funcionales que ayudan a establecer el polinomio de interpolación. Este método fué diseñado para utilizar técnicas recursivas aunque no necesariamente para el uso de una computadora.

Supongamos que $p_n(x)$ es un polinomio de grado a lo más n que coincide con la función f en los puntos $x_0, x_1, \ldots, x_n$ (todos los puntos distintos).

Una forma de representar a $p_n(x)$ es:

$$p_n(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \ldots + c_n(x - x_0)(x - x_1)(x - x_2)\ldots(x - x_{n-1}) \quad (**)$$

siendo c_i valores constantes $i = 0, 1, \ldots, n$.

Ahora, para encontrar la constante c_0 se evalua (**) en el primer punto x_0 ; éste es únicamente el término constante c_0,

$$c_0 = p_n(x_0) = f(x_0).$$

De la misma manera, al evaluar el siguiente punto x_1, los términos diferentes de cero son el término constante c_0 y el término lineal

$$f(x_0) + c_1(x_1 - x_0) = p_n(x_1) = f(x_1)$$

de esta manera

$$c_1 = \frac{f(x_1) - f(x_0)}{(x_1 - x_0)}$$

y llegamos a la notación de lo que es la *segunda diferencia dividida*.

Entonces, inicialmente, la diferencia dividida cero es la función f evaluada en el punto x_0 y queda de la siguiente manera

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

y en lo que respecta a las siguientes, las definimos inductivamente. Así, la diferencia dividida de x_i y x_{i+1} que se denota como $f[x_i, x_{i+1}]$ está dada por

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}$$

Cuando han sido encontradas las primeras $(k-1)$ diferencias divididas $f[x_i, x_{i+1}, x_{i+2}, \ldots, x_{i+k-1}]$ y $f[x_{i+1}, x_{i+2}, \ldots, x_{i+k-1}, x_{i+k}]$, entonces ha sido determinada la k-ésima diferencia dividida bajo f en los puntos $x_i, x_{i+1}, x_{i+2}, \ldots, x_{i+k}$ quedando así

$$f[x_i, x_{i+1}, x_{i+2}, \ldots, x_{i+k-1}, x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, \ldots, x_{i+k}] - f[x_i, x_{i+1}, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$$

para $i = 0, 1, \ldots, n$

Entonces la expresión (**) se puede representar nuevamente

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

Esto equivale a decir que las constantes c_i para $i = 0, \ldots, n$ tienen una representación de la siguiente manera

$$c_0 = f[x_0]$$
$$c_1 = f[x_0, x_1]$$
$$c_2 = f[x_0, x_1, x_2]$$
$$\vdots$$
$$c_n = f[x_0, x_1, x_2, \ldots, x_n]$$

de esta forma para cualquier valor k, $0 \le k \le n$; $c_k = f[x_0, x_1, \ldots, x_k]$.

La expresión (***) puede ser representada así

$$p_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, \ldots, x_k](x - x_0)\cdots(x - x_{k-1})$$

y ésta es la fórmula de Newton para interpolación, donde $c_k = f[x_0, x_1, \ldots, x_k]$ es la k-ésima diferencia dividida y también es el coeficiente de x^{k-1} en el polinomio p_{k-1}.

III. Factorización de Matrices

Consideramos un sistema no-homogéneo de n ecuaciones lineales algebraicas en "n" variables, $x_1, x_2, \ldots, x_n$; simbólicamente de la siguiente forma $A\mathrm{x} = \mathrm{b}$

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ \vdots \qquad\qquad\qquad\quad & \vdots \quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned}$$

que en términos de matrices tiene una representación

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

donde

$A = (a_{ij})$ es la matriz de coeficientes

$x = (x_i)$ es el vector de incógnitas

$b = (b_i)$ es el vector de constantes

Si el determinante de la matriz es diferente de cero, $\det A \neq 0$, entonces existe una solución única x de dicho sistema de ecuaciones.
Por lo tanto hay una única matriz inversa A^{-1} tal que

$$A^{-1}Ax = A^{-1}b$$

y como $A^{-1}A = 1$

y se obtiene la solución $x = A^{-1}b$

A éste sistema $A\mathrm{x} = \mathrm{b}$ se le da solución mediante dos sistemas triangulares. Decimos que la matriz A tiene factorización LU, cuando se puede llevar a una forma escalonada (para que luego se pueda efectuar la sustitución regresiva); éste método de triangularización también es conocido como *eliminación Gaussiana* sin intercambio de renglones o columnas, necesario ésto, para llegar a la última etapa que es la solución $U\mathrm{x} = \mathrm{c}$, $L\mathrm{c} = \mathrm{b}$ y donde $A = LU$, L es una matriz triangular inferior y U es una matriz triangular superior.
Así, el sistema $A\mathrm{x} = \mathrm{b}$ queda $LU\mathrm{x} = L\mathrm{c} = \mathrm{b}$.

La matriz A es no-singular $|A| \neq 0$, y tiene igual número de elementos en filas y columnas.

Consideramos la matriz triangular inferior $L = (l_{ij})$ definida como aquella matriz que para cada

$$l_{ij} = \begin{cases} 0, & \text{cuando } i = 1,2,\ldots,\text{j-1} \\ 1, & \text{cuando } i = j \\ m_{ij}, & \text{cuando } i = j+1,\text{j}+2,\ldots,n \end{cases}$$

donde el elemento i,j de la matriz final calculada se llega a el mediante el método de eliminación gaussiana y m_{ij} es el multiplicador

$$L = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -m_{21} & 1 & 0 & \cdots & 0 \\ -m_{31} & -m_{32} & 1 & & 0 \\ \cdot & \cdot & \cdot & \ddots & \\ -m_{n1} & -m_{n2} & \cdots & -m_{nn-1} & 1 \end{bmatrix}$$

Denotamos por $e_1, e_2, \ldots, e_n$, las operaciones elementales entre filas, y E_i las matrices asociadas a esas operaciones, para $i = 1,2,\ldots,n$

Consideramos la matriz triangular superior $U = (u_{ij})$ de $n \times n$ definida como cierta matriz que tiene ceros por debajo de la diagonal principal es decir , para cada

$$u_{ij} = \begin{cases} 0, & \text{cuando } i = j+1,\text{j}+2,\ldots,n \\ a_{ij}, & \text{cuando } i = 1,2,\ldots,j \end{cases}$$

Una matriz diagonal es a la vez triangular superior e inferior. Estas forman parte de los tipos especiales de matrices.

$$U=\begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & & u_{3n} \\ 0 & 0 & \cdot & \ddots & \cdot \\ 0 & 0 & \cdots & 0 & u_{nn} \end{bmatrix}$$

Supongamos que V es una matriz no singular que puede llevarse a la forma triangular (superior) U mediante el uso de operaciones elementales de filas , V puede triangularizarse.

Las operaciones elementales entre filas en el algoritmo de triangularización son las que denotamos anteriormente como $e_1, e_2, \ldots, e_n$, las inversas de dichas operaciones elementales en orden inverso sobre la matriz identidad I genera la matriz L

$$L = E_1^{-1} E_2^{-1} \cdots E_n^{-1} I$$

donde $E_1, \ldots, E_n$ son las matrices asociadas a las operaciones elementales e_i; $i = 1, 2, \ldots, n$. Por otro lado éstas mismas operaciones elementales $e_1, e_2, \ldots, e_n$, transforman la matriz original V en una matriz triangular superior U

$$E_n E_{n-1} \cdots E_2 E_1 V = U$$

Ahora, despejando la matriz V en la expresión anterior llegamos a :

$$V = \left(E_1^{-1} E_2^{-1} \cdots E_n^{-1}\right) U = \left(E_1^{-1} E_2^{-1} \cdots E_n^{-1} I\right) U = LU$$

que no es otra cosa que la descomposición "L U" de la matriz V, que en definitiva se redujo a dos sistemas triangulares: a una sustitución directa ($Lc = b$) y a otra regresiva ($Ux = c$). Lo anterior solo se aplica a matrices no singulares y que puedan triangularizarse sin necesidad de intercambiar filas. Es cuando se nombran las matrices *factorizables LU*.

A continuación damos un ejemplo: Resolver el sistema:

$$\begin{bmatrix} 2 & 1 & 1 \\ 4 & 1 & 0 \\ -2 & 2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 7 \end{bmatrix}$$

A través de las siguientes operaciones elementales se obtiene la matriz triangular superior U

1.- Restar al 2o. renglón el 1o. multiplicado por 2.

2.- Restar al 3o. renglón el 1o. multiplicado por -1.

3.- Restar al 3o. renglón el 2o. multiplicado por -3.

Una vez que se aplicaron estos pasos, el resultado es

$$Ux = \begin{bmatrix} 2 & 1 & 1 \\ 0 & -1 & -2 \\ 0 & 0 & -4 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ -4 \\ -4 \end{bmatrix}$$

Hasta aquí cabe decir que queda una sustitución regresiva.

Veamos ahora que mediante las matrices elementales, haciendo uso de los multiplicadores (en al paso No.1 fué el 2; en al paso No.2 fué el -1; y en al paso No.3 fué el -3. De otra manera $m_{12} = 2, m_{13} = -1, m_{23} = -3$. Y utilizando una matriz identidad I, se procede :

Se sustituye el cero por el multiplicador intercambiándole los subíndices y cambiando el signo

$m_{12} = 2$, la matriz elemental es $E_{21} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ contiene al elemento $-m_{21} = 2$

$m_{13} = -1$, la matriz elemental es $E_{31} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$ contiene al elemento $-m_{31} = -1$

$m_{23} = -3$, la matriz elemental es $E_{32} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 3 & 1 \end{bmatrix}$ contiene al elemento $-m_{32} = -3$

Las tres operaciones elementales que convierten a V en U representadas en forma de matrices, son $E_{32}E_{31}E_{21}V = U$, al igual con los términos no homogéneos $E_{32}E_{31}E_{21}\mathrm{b} = \mathrm{c}$. Sería bueno tener una sola matriz E que convierta de una sola vez a V y b en U y c respectivamente.

Esta matriz es $E_{32}E_{31}E_{21} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -5 & 3 & 1 \end{bmatrix}$

y para regresar de la matriz U a la matriz V, se hace lo siguiente: si en la operación elemental e_1 de la matriz E_{21} se restó al segundo renglón el doble del primero, pues ahora se le suma al segundo, el doble del primero. Se inicia poniendo la matriz identidad primeramente y efectuando operaciones elementales

$$E_{21}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} E_{31}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} E_{32}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{bmatrix}$$

Haciendo el producto $E_{21}E_{21}^{-1} = I$, cada matriz elemental tiene su inversa. De la misma forma, la segunda y tercera matrices elementales pueden invertirse sumando lo que fué restado en los pasos ii) y iii).

Regla general: las inversas vienen en el orden opuesto a la sucesión original de operaciones.

$$V = E_{21}^{-1}E_{31}^{-1}E_{32}^{-1}U$$

Se puede sustituir $U = E_{32}E_{31}E_{21}V$ en esta ecuación y ver que está correcto el orden de las matrices. Entonces la matriz L que lleva a U de regreso a V tiene que ser el producto de las matrices que invierten cada paso individual

$$L = E_{21}^{-1}E_{31}^{-1}E_{32}^{-1}$$

Así que $V = LU$ y esta matriz L es la clave de la Eliminación Gaussiana; es el vínculo entre la matriz V con la que comenzamos y la matriz U a la que llegamos.

Siendo L una matriz triangular inferior, vemos que por abajo de la diagonal principal están los multiplicadores 2,-1 y -3 usados en los tres pasos de la eliminación gaussiana. Obtenemos :

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -3 & 1 \end{bmatrix} \quad \text{y} \quad U = \begin{bmatrix} 2 & 1 & 1 \\ 0 & -1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

IV. Método de Horner

Cuando en el método de Horner se hacen los cálculos a mano, se construye una tabla, usando la "división sintética", En el presente trabajo, siendo $P(x)=a_n x^n+a_{n-1}x^{n-1}+\cdots+a_1 x+a_0$ un polinomio de grado n, obtenemos los coeficientes a_i, constantes del polinomio P.

En dicho método, se construyen polinomios $f_0(x), f_1(x), f_2(x),\ldots,$. Una vez conocido $f_i(x)$, puede obtenerse a_{i+1} , y una vez obtenida a_{i+1} podemos obtener $f_{i+1}(x)$. El método queda descrito una vez que se describan cómo obtener : $f_i(x)$, a_{i+1} conocido $f_i(x)$ y, $f_{i+1}(x)$ conocidos $f_i(x)$ y a_{i+1}. Pensemos en un polinomio $f(x)$ de grado n y un complejo a y hagamos la siguiente serie de divisiones :

$$f(x)=(x-a)f_1(x)+b_0$$
$$f_1(x)=(x-a)f_2(x)+b_1$$
$$\bullet\ \bullet\ \bullet$$
$$f_{n-1}(x)=(x-a)f_n(x)+b_{n-1}$$

entonces el polinomio puede expresarse en la forma $f(x)=\sum_{i=0}^{n} b_i(x-a)^i$, en donde b_0 es el resto de la división de $f(x)$ entre $x-a$; b_1 el resto de la división del cociente de esa primera división entre $x-a$; b_2 el resto de la división del nuevo cociente entre $x-a$, etc.

TEOREMA (Método de Horner) Sea

$$P(x)=a_n x^n+a_{n-1}x^{n-1}+\cdots+a_1 x+a_0 \qquad \text{y} \qquad b_n=a_n$$

si

$$b_k=a_k+b_{k+1}x_0 \qquad \text{para} \quad k=n-1,n-2,\ldots,1,0,$$

entonces $b_0=P(x_0)$. Además, si

$$Q(x)=b_n x^{n-1}+b_{n-1}x^{n-2}+\cdots+b_2 x+b_1 \ ,$$

entonces

$$P(x)=(x-x_0)Q(x)+b_0$$

Demostración

Por definición $(x-x_0)Q(x)+b_0=(x-x_0)\left(b_nx^{n-1}+b_{n-1}x^{n-2}+\cdots+b_2x+b_1\right)+b_0$

$$=b_nx^n+b_{n-1}x^{n-1}+\cdots+b_2x^2+b_1x-(b_nx_0x^{n-1}+\cdots+b_2x_0x+b_1x_0)+b_0$$

$$=b_nx^n+(b_{n-1}-b_nx_0)x^{n-1}+\cdots+(b_1-b_2x_0)x+(b_0-b_1x_0)$$

Por hipótesis del teorema,

$b_n=a_n$ y $a_k=b_k-b_{k+1}x_0$ así que

$(x-x_0)Q(x)+b_0=P(x)$ y $b_0=P(x_0)$.

CAPITULO 1

CASO NO-CONFLUENTE

1.1 Algoritmos

1.1.1. Dual

A fin de darle solución a un sistema de ecuaciones lineales simultáneas no homogéneas

$$\begin{aligned} x_0 + \quad x_1 + \cdots + \quad x_n &= b_0 \\ \alpha_0 x_0 + \alpha_1 x_1 + \cdots + \alpha_n x_n &= b_1 \\ \alpha_0^2 x_0 + \alpha_1^2 x_1 + \cdots + \alpha_n^2 x_n &= b_2 \\ \vdots \qquad\qquad\qquad \vdots \quad \vdots & \\ \alpha_0^n x_0 + \alpha_1^n x_1 + \cdots + \alpha_n^n x_n &= b_n \end{aligned}$$

con matriz, de coeficientes tipo Vandermonde, describimos a continuación el algoritmo desarrollado por A.Björck y V. Pereyra [2].

Sean $\alpha_0,\ldots,\alpha_n \in \mathbb{R}$ (éstos son puntos ó nodos), n un número natural y V una matriz en $\mathbb{R}^{(n+1)\times(n+1)}$; denotamos de la siguiente forma el vector columna $V(\alpha) = V(1,\alpha,\alpha^2\ldots,\alpha^n)^T$ de esa matriz evaluada en un punto α,

$$V = V(\alpha_0,\alpha_1,\ldots,\alpha_n) = \begin{vmatrix} 1 & 1 & & 1 \\ \alpha_0 & \alpha_1 & \cdots & \alpha_n \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_0^n & \alpha_1^n & \cdots & \alpha_n^n \end{vmatrix} = \left[V(\alpha_0),\ldots,V(\alpha_n),\right]$$

así presentamos la matriz de Vandermonde. Y probamos que V es no-singular

Si $\det V_n = \begin{vmatrix} 1 & \alpha_0 & \alpha_0^2 & \cdots & \alpha_0^{n-1} \\ 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{n-1} \\ \vdots & & & & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \cdots & \alpha_n^{n-1} \end{vmatrix}$ luego $V_n = (\alpha_n - \alpha_0)(\alpha_{n-1} - \alpha_0)\cdots(\alpha_2 - \alpha_0)V_{n-1}$.

Utilizando las propiedades de los determinantes hacemos ceros en el primer renglón empezando con la tercera columna y hasta el final del primer renglón, esto para reducir el orden del determinante y facilitar las operaciones.

$$\det V_n = \begin{vmatrix} 1 & \alpha_0 & 0 & & 0 \\ 1 & \alpha_1 & \alpha_1^2 - \alpha_0\alpha_1 & \cdots & \alpha_1^{n-1} - \alpha_0\alpha_1^{n-2} \\ \vdots & & & & \vdots \\ 1 & \alpha_n & \alpha_n^2 - \alpha_0\alpha_n & \cdots & \alpha_n^{n-1} - \alpha_0\alpha_n^{n-2} \end{vmatrix}$$, luego hacemos cero en la 2a.

columna y renglón primero : $\det V_n = \begin{vmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & \alpha_1 - \alpha_0 & \alpha_1^2 - \alpha_0\alpha_1 & \cdots & \alpha_1^{n-1} - \alpha_0\alpha_1^{n-2} \\ \vdots & & & & \vdots \\ 1 & \alpha_n - \alpha_0 & \alpha_n^2 - \alpha_0\alpha_n & \cdots & \alpha_n^{n-1} - \alpha_0\alpha_n^{n-2} \end{vmatrix}$

de esta manera podemos eliminar el primer renglón de la matriz, quedando :

$$V_n = (\alpha_1 - \alpha_0)(\alpha_2 - \alpha_0)(\alpha_3 - \alpha_0)\cdots(\alpha_n - \alpha_0)\begin{vmatrix} 1 & \alpha_0 & \alpha_0^2 & \cdots & \alpha_0^{n-2} \\ 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{n-2} \\ \vdots & & & & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \cdots & \alpha_n^{n-2} \end{vmatrix}$$, y esta matriz es :

$$V_{n-1} = \begin{vmatrix} 1 & \alpha_0 & \alpha_0^2 & \cdots & \alpha_0^{n-2} \\ 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{n-2} \\ \vdots & & & & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \cdots & \alpha_n^{n-2} \end{vmatrix} \therefore \det V_n = (\alpha_1 - \alpha_0)(\alpha_2 - \alpha_0)(\alpha_3 - \alpha_0)\cdots(\alpha_n - \alpha_0)\det V_{n-1}$$

Por inducción 1) $n = 1$; $\det V_1 = (\alpha_1 - \alpha_0)$

2) $n = n+1$; $\det V_2 = (\alpha_2 - \alpha_0)\det V_{n-1}$; lo que prueba la validez de la fórmula $V_n = \prod_{j>i}(\alpha_j - \alpha_i)$ para $(0 \le i, j \le n)$. Por lo tanto $V_n \neq 0$

Ahora consideramos los siguientes dos sistemas de ecuaciones lineales:

(1) $V\,\mathrm{x} = \mathrm{b}$

(2) $V^T\,\mathrm{a} = \mathrm{f}$

Resolvemos tanto el (1) que es el sistema directo de Vandermonde ó *Primal*:

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_0 & \alpha_1 & \cdots & \alpha_n \\ \vdots & \vdots & \cdots & \vdots \\ \alpha_0^n & \alpha_1^n & \cdots & \alpha_n^n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}$$

como el (2) que es el sistema *Dual* de Vandermonde

$$\begin{bmatrix} 1 & \alpha_0 & \alpha_0^2 & \cdots & \alpha_0^n \\ 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^n \\ \vdots & & & & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \cdots & \alpha_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix}.$$

Resolvemos primero el caso No-confluente, para el sistema *Dual* o sea $\alpha_i \neq \alpha_j$ para $i \neq j$

Los sistemas Vandermonde se fundamentan en problemas de aproximación e interpolación. Por tal motivo hacemos uso de la interpolación polinomial.

A continuación resolvemos el sistema *Dual* $V^T \mathrm{a} = \mathrm{f}$. Aplicando el producto de vectores y matrices queda

$$\begin{aligned} &a_0 + a_1\alpha_0 + a_2\alpha_0^2 + \cdots + a_n\alpha_0^n = f_0 \\ &a_0 + a_1\alpha_1 + a_2\alpha_1^2 + \cdots + a_n\alpha_1^n = f_1 \\ &a_0 + a_1\alpha_2 + a_2\alpha_2^2 + \cdots + a_n\alpha_2^n = f_2 \\ &\vdots \qquad\qquad \vdots \qquad \vdots \qquad\qquad (*) \\ &a_0 + a_1\alpha_n + a_2\alpha_n^2 + \cdots + a_n\alpha_n^n = f_n \end{aligned}$$

Definimos el polinomio

$$p(\alpha) = \sum_{j=0}^{n} a_j \alpha^j$$

y observamos que todo el sistema (*) implica que $p(\alpha_i) = f_i$ para $i = 0, 1, \ldots, n$.

Teniendo en cuenta siempre que todas las α_i son distintas, entonces existe un único polinomio de grado n que interpola a los diferentes puntos (sección I.). Encontramos las a_k mediante el método de interpolación polinomial de Newton.

Recordando: $P(x)=\sum_{k=0}^{n} a_k \prod_{i=0}^{k-1}(x-\alpha_i)$

Damos inicio introduciendo el siguiente polinomio. Escribimos $P(x)$ como :

$$P(x)=a_0+a_1x+a_2x^2+\cdots+a_nx^n$$

Para determinar los coeficientes definimos

$$q_0(x)=1 \qquad \text{y} \qquad q_k(x)=\prod_{i=0}^{k-1}(x-\alpha_i) \quad \text{para} \quad k=1,2,\ldots n$$

desarrollando la fórmula anterior

$$\begin{aligned}
q_0(x)&=1\\
q_1(x)&=(x-\alpha_0)\\
q_2(x)&=(x-\alpha_0)(x-\alpha_1)\\
q_3(x)&=(x-\alpha_0)(x-\alpha_1)(x-\alpha_2)\\
&\vdots\\
q_n(x)&=(x-\alpha_0)(x-\alpha_1)(x-\alpha_2)\ldots(x-\alpha_{n-1})
\end{aligned}$$

entonces para evaluar el polinomio $P(x)=(q_0(x),q_1(x),\ldots,q_n(x))\mathrm{c}$ donde $\mathrm{c}=(c_0,c_1,\ldots,c_n)$ o también $c_0q_0(x)+c_1q_1(x)+\ldots+c_nq_n(x)$ hay que involucrar todos los polinomios como se observa en $P(x)$.

$$P(x)=c_0+c_1\left[(x-\alpha_0)\right]+c_2\left[(x-\alpha_0)(x-\alpha_1)\right]+\ldots+c_n\left[(x-\alpha_0)(x-\alpha_1)\ldots(x-\alpha_{n-1})\right]$$

Traducido a la fórmula de Newton para interpolación polinomial

$$P(x)=\sum_{k=0}^{n} c_k \prod_{i=0}^{k-1}(x-\alpha_i)$$

donde c_k para $k=0,1,\ldots,n$ son las *Diferencias Divididas* de orden k, mas claro;

$$c_k=f\left[\alpha_0,\alpha_1,\ldots,\alpha_k\right]$$

para recordar $c_0 = f[\alpha_0]$

$c_1 = f[\alpha_0, \alpha_1]$

$c_2 = f[\alpha_0, \alpha_1, \alpha_2]$

$\vdots$

$c_k = f[\alpha_0, \alpha_1, \ldots, \alpha_k]$

Una observación es que $f[\alpha_0, \alpha_1, \alpha_2]$ es el coeficiente de x^2 en el polinomio cuadrático que interpola f en los puntos $\alpha_0, \alpha_1, \alpha_2$. En un sentido más general, esto quiere decir, que cada una de las diferencias divididas representa el coeficiente de x^{k-1} en el polinomio P_{k-1} de grado $\leq k-1$ que interpola f en los puntos $x_0, \ldots, x_k$. ,

$$f[\alpha_0] = f(\alpha_0)$$

$$f[\alpha_0, \alpha_1] = \frac{f[\alpha_1] - f[\alpha_0]}{\alpha_1 - \alpha_0}$$

$$f[\alpha_0, \alpha_1, \alpha_2] = \frac{f[\alpha_1, \alpha_2] - f[\alpha_0, \alpha_1]}{\alpha_2 - \alpha_0}$$

La manera de representar la recursividad es:

$$f[\alpha_{j-k-1}, \ldots, \alpha_j] = \frac{f[\alpha_{j-k}, \ldots, \alpha_j] - f[\alpha_{j-k-1}, \ldots, \alpha_{j-1}]}{\alpha_j - \alpha_{j-k-1}}$$

y ésta es la fórmula general.

Para evaluar la interpolación polinomial la expresión es

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

que en forma compacta queda : $P_n(x) = \sum_{k=0}^{n} f[x_0, \ldots, x_k] \prod_{i=0}^{k-1} (x - x_i)$

Así por ejemplo, si $n = 1$, el polinomio a evaluar es $p_1(x) = f[x_0] + f[x_0, x_1](x - x_0)$.

Luego entonces como primer paso, procedemos a obtener las diferencias divididas para el sistema de Vandermonde

$$\begin{bmatrix} 1 & \alpha_0 & \alpha_0^2 & \cdots & \alpha_0^n \\ 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^n \\ \vdots & & & & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \cdots & \alpha_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix}$$

tomamos los valores α_i para $i = 0,1,\ldots,n$ y las f_i.

Iniciando...(por interpolación polinomial)

$$q_0(x) = f[\alpha_0] = f(\alpha_0)$$

$$q_1(x) = f[\alpha_0] + f[\alpha_0,\alpha_1](\alpha_1 - \alpha_0)$$

$$f[\alpha_1] - f[\alpha_0] = f[\alpha_0,\alpha_1](\alpha_1 - \alpha_0)$$

$$\frac{f[\alpha_1] - f[\alpha_0]}{\alpha_1 - \alpha_0} = f[\alpha_0,\alpha_1]$$

o sea $f(\alpha_2) = q_2(x) = f[\alpha_0] + f[\alpha_0,\alpha_1](\alpha_2 - \alpha_0) + f[\alpha_0,\alpha_1,\alpha_2](\alpha_2 - \alpha_0)(\alpha_2 - \alpha_1)$

entonces $f(\alpha_2) - f(\alpha_0) - f[\alpha_0,\alpha_1](\alpha_2 - \alpha_0) = f[\alpha_0,\alpha_1,\alpha_2](\alpha_2 - \alpha_0)(\alpha_2 - \alpha_1)$, y se

sigue $f(\alpha_2) - f(\alpha_0) - \frac{f(\alpha_1) - f(\alpha_0)}{\alpha_1 - \alpha_0}(\alpha_2 - \alpha_0) = f[\alpha_0,\alpha_1,\alpha_2](\alpha_2 - \alpha_0)(\alpha_2 - \alpha_1)$,

despejando el término, llegamos a la tercera diferencia dividida, quedando como

$$f[\alpha_0,\alpha_1,\alpha_2] = \frac{f(\alpha_2) - f(\alpha_0) - \frac{f(\alpha_1) - f(\alpha_0)}{(\alpha_1 - \alpha_0)}(\alpha_2 - \alpha_0)}{(\alpha_2 - \alpha_0)(\alpha_2 - \alpha_1)}$$

Hasta aquí se ha obtenido la tercera diferencia dividida, pero para el caso en que sea necesario obtener hasta n-diferencias divididas, se hace mediante un algoritmo que ejecute éstos cálculos. En el algoritmo que a continuación se ilustra, la notación $c^{(0)} = f$ indica que el término independiente (en el sistema DUAL $V^T a = f$), se almacena a partir de la primera posición en el vector $c^{(0)}$.

Paso No.1 (Diferencias Divididas)

$c^{(0)} = f$

For $k = 0,1,\ldots,n-1$

For $j = n, n-1, \ldots, k+1$

$$c_j^{(k+1)} = \left(c_j^{(k)} - c_{j-1}^{(k)}\right) \Big/ \left(\alpha_j - \alpha_{j-k-1}\right)$$

Una vez que se han conocido las c's o diferencias divididas, a partir de éstas procedemos al cálculo de los coeficientes a's. Esto requiere de efectuar las operaciones para evaluar cada término, y es lo que se llama esquema de Horner, el cual nos permite evaluar los polinomios $(q_0(x), q_1(x), \ldots, q_n(x),)$ los cuales son presentados en forma usual. Así, agregando uno a uno incluímos todos los términos de los polinomios involucrados en $P(x)$. Denotamos dichos polinomios con los símbolos $q_n(x), \ldots, q_o(x)$ en el siguiente algoritmo recursivo.

$q_n(x) = c_n$

For $k = n-1, n-2, \ldots, 1, 0$

$q_k(x) = c_k + (x - \alpha_k) q_{k+1}(x)$

Aquí se observa que cuando se evalúa el polinomio $q_0(x)$, es porque ya terminó todo el procedimiento recursivo, y ya contempla todos los polinomios involucrados, es decir $P(x) = q_0(x)$.

A continuación mediante operaciones algebraicas observamos que escribiendo $q_k(x) = a_k^{(k)} + a_{k+1}^{(k)} x + \cdots + a_n^{(k)} x^{n-k}$ e igualando las potencias de x en la ecuación $q_k(x) = c_k + (x - \alpha_k) q_{k+1}(x)$ obtenemos los coeficientes a_i^k.

Iniciamos...

$$c_k + (x - \alpha_k) q_{k+1}(x) = a_k^{(k)} + a_{k+1}^{(k)} x + \cdots + a_n^{(k)} x^{n-k}$$

el polinomio $q_{k+1}(x) = a_{k+1}^{(k+1)} + a_{k+2}^{(k+1)} x + \cdots + a_n^{(k+1)} x^{n-(k+1)}$ lo sustituimos en el renglón de arriba quedando una expresión como:

$$c_k + (x - \alpha_k)\left[a_{k+1}^{(k+1)} + a_{k+2}^{(k+1)} x + \cdots + a_n^{(k+1)} x^{n-k-1}\right] = a_k^{(k)} + a_{k+1}^{(k)} x + \cdots + a_n^{(k)} x^{n-k}$$

realizando el producto del paréntesis por el corchete, tenemos:

$$c_k + (x - \alpha_k) a_{k+1}^{(k+1)} + (x - \alpha_k) a_{k+2}^{(k+1)} x + \ldots (x - \alpha_k) a_n^{(k+1)} x^{n-k-1} = a_k^{(k)} + a_{k+1}^{(k)} x + \ldots + a_n^{(k)} x^{n-k}$$

desarrollando el paréntesis

$$c_k + x a_{k+1}^{(k+1)} - \alpha_k a_{k+1}^{(k+1)} + x a_{k+2}^{(k+1)} x - \alpha_k a_{k+2}^{(k+1)} x + \ldots + x a_n^{(k+1)} x^{n-k-1} - \alpha_k a_n^{(k+1)} x^{n-k-1} =$$

$$= a_k^{(k)} + a_{k+1}^{(k)}x + \ldots + a_n^{(k)}x^{n-k}$$

reacomodando las potencias de x

$$c_k + a_{k+1}^{(k+1)}x - \alpha_k a_{k+1}^{(k+1)} + a_{k+2}^{(k+1)}x^2 - \alpha_k a_{k+2}^{(k+1)}x + \ldots + a_n^{(k+1)}x^{n-k} - \alpha_k a_n^{(k+1)}x^{n-k-1} = a_k^{(k)} + a_{k+1}^{(k)}x + \ldots + a_n^{(k)}x^{n-k}$$

reagrupando el término independiente.

$$c_k - \alpha_k a_{k+1}^{(k+1)} = a_k^{(k)}$$

el témino lineal

$$a_{k+1}^{(k+1)}x - \alpha_k a_{k+2}^{(k+1)}x = a_{k+1}^{(k)}x$$

o también

$$a_{k+1}^{(k+1)} - \alpha_k a_{k+2}^{(k+1)} = a_{k+1}^{(k)}$$

y así sucesivamente hasta la última potencia de x

$$a_n^{(k+1)}x^{n-k} = a_n^{(k)}x^{n-k}$$

o bien

$$a_n^{(k+1)} = a_n^{(k)}$$

Entonces mediante la recurrencia podemos computar las desconocidas $a_k = a_k^{(0)}$.

Ahora introducimos para $k = 0, 1, \ldots, n$, los siguientes vectores

$$\mathrm{c}^{(k)} = \left(c_0, \ldots, c_k, f\left[\alpha_1, \ldots, \alpha_{k+1}\right], f\left[\alpha_{n-k}, \ldots, \alpha_n\right]\right)^T$$

y

$$\mathrm{a}^{(k)} = \left(c_0, \cdots, c_{k-1}, a_k^{(k)}, \cdots a_n^{(k)}\right)^T$$

De éstas definiciones observamos de inmediato que

$$\mathrm{c}^{(0)} = \mathrm{f}, \quad \mathrm{c}^{(n)} = \mathrm{a}^{(n)} = \mathrm{c}, \quad \mathrm{a}^{(0)} = \mathrm{a}$$

y de las diferencias divididas, esquema de Horner y polinomio, conseguimos finalmente llegar al algoritmo Dual.

Paso No.2 (Esquema de Horner)

$a^{(n)} = c$

For $k = n-1,\ldots 1,0$

For $j = k, k+1,\ldots,n-1$

$$a_j^{(k)} = a_j^{(k)} - \alpha_k a_{j+1}^{(k+1)}$$

Entonces los pasos No. 1 y el No. 2 juntos constituyen el procedimiento que da la solución al sistema Dual.

RESUMEN :

Dados distintos escalares reales $\alpha_0,\ldots,\alpha_n$ y $f = (f_0,\ldots f_n)^T \in \mathbb{R}^{n+1}$ el siguiente algoritmo proporciona en f la solución $(a_0,\ldots a_n)^T$ para un sistema de Vandermonde $V(\alpha_0,\ldots,\alpha_n)^T a = f$

ALGORITMO DUAL

For $k = 0,\ldots,n-1$

For $j = n,\ldots,k+1$

$$f_j = \left(f_j - f_{j-1}\right) \Big/ \left(\alpha_j - \alpha_{j-k-1}\right)$$

For $k = n-1,\ldots,0$

For $j = k,\ldots,n-1$

$$a_j = f_j$$

$$a_j = a_j - a_{j+1}\alpha_k$$

1.1 Algoritmos

1.1.2. Primal

Ahora para darle solución al sistema $V\,\mathrm{x} = \mathrm{b}$ en el lenguaje de vectores y matrices, introducimos una matriz bidiagonal elemental definida así:

$L_k(\alpha) \equiv l_{st}$ para $0 \le k \le n-1$ dada por la regla

$$l_{st} \equiv \begin{cases} 1 & \text{si} \quad s = t \\ -\alpha & \text{si} \quad s = k+2,\ldots,n; \quad t = s-1 \\ 0 & \text{otro} \end{cases}$$

Esta matriz es una bidiagonal inferior de tamaño $(n+1)\times(n+1)$ que toma la siguiente forma

$$L_k(\alpha) = \left[\begin{array}{c|cc} I_k & \multicolumn{2}{c}{0} \\ \hline & 1 & \\ 0 & -\alpha & 1 \\ & & -\alpha \quad 1 \end{array}\right]$$

donde I_k es la matriz identidad de tamaño k.

Definimos tambien una matriz diagonal del siguiente modo:

$$D_k = \mathrm{diag}\left(\underbrace{1,\ldots,1}_{k+1},(\alpha_{k+1}-\alpha_0),\ldots,(\alpha_n-\alpha_{n-k-1})\right)$$

Con éstas dos definiciones es fácil verificar mediante el Paso No.1 del algoritmo Dual que los vectores $\mathrm{c}^{(k)}$ son recursivamente obtenidos así:

$$\mathrm{c}^{(0)} = \mathrm{f}, \qquad \mathrm{c}^{(k+1)} = D_k^{-1} M_k \mathrm{c}^{(k)}, \qquad \text{para} \quad k = 0,1,\ldots,n-1$$

donde $M_k = L_k(1)$. Entonces si $\mathrm{f} = (f_0,\ldots f_n)^{\mathrm{T}}$ y $\mathrm{c} = (c_0,\ldots,c_n)^{\mathrm{T}}$

$$\mathrm{c} = U^{\mathrm{T}}\mathrm{f}$$

dónde U^T es la matriz triangular superior definida por

$$U^{\mathrm{T}} = D_{n-1}^{-1} M_{n-1} \cdots D_0^{-1} M_0$$

Y ahora se verifica mediante el Paso No. 2,

$$\mathrm{a}^{(n)} = \mathrm{c}, \qquad \mathrm{a}^{(k)} = N_k^T \mathrm{a}_k^{(k+1)} \qquad \text{para } k = n-1,\ldots,1,0$$

donde $N_k = L_k(\alpha_k)$

Esta segunda parte plantea:

$$\mathrm{a} = L^T \mathrm{c}$$

donde L es la matriz triangular inferior definida por

$$L^T = N_0^T N_1^T \cdots N_{n-1}^T.$$

Juntando la primera parte $\mathrm{c} = U^T \mathrm{f}$, con la segunda $\mathrm{a} = L^T \mathrm{c}$, resulta

$$\mathrm{a} = L^T U^T \mathrm{f}$$

Comparándo esto con el Dual $V^T \mathrm{a} = \mathrm{f}$, ó $\mathrm{a} = (V^{-1})^T \mathrm{f}$, de donde vemos claramente que $(V^{-1})^T = L^T U^T$, ó también $V^{-1} = UL$. En otras palabras se dice que el algoritmo Primal se resuelve tácticamente mediante la descomposición UL de la matriz V^{-1}.

Luego, la factorización de V^{-1} puede ser usada para describir un algoritmo para resolver $V\mathrm{x} = \mathrm{b}$; veamos...

$$\mathrm{x} = V^{-1}\mathrm{b} = \left(M_0^T D_0^{-1} \cdots M_{n-1}^T D_{n-1}^{-1}\right)\left(N_{N-1} \cdots N_1 N_0\right) b$$

ésta es una relación de recurrencia para calcular $\mathrm{x} = \mathrm{x}^{(0)}$,

$$\mathrm{d}^{(0)} = \mathrm{b}, \qquad \mathrm{d}^{(k+1)} = N_k \mathrm{d}^{(k)}, \qquad \text{para } k = 0,1,\ldots,n-1$$

$$\mathrm{x}^{(n)} = \mathrm{d}^{(n)}, \qquad \mathrm{x}^{(k)} = M_k^T D_k^{-1} \mathrm{x}^{(k+1)}, \qquad \text{para } k = n-1,\ldots,1,0$$

RESUMEN: Dados distintos escalares reales $\alpha_0,\ldots,\alpha_n$ y $\mathrm{b} = (b_0,\ldots b_n)^T \in \mathbb{R}^{n+1}$, obtenemos la solución $\mathrm{x} = (x_0,\ldots x_n)^T$, para el sistema de Vandermonde $V\mathrm{x} = \mathrm{b}$.

ALGORITMO PRIMAL

For $i = 0,\ldots,n$; $x_i = b_i$

For $k = 0,\ldots,n-1$

  For $i = n,\ldots,k+1$

    $x_i = x_i - x_{i-1}\alpha_k$

For $k = n-1,\ldots,0$

  For $i = k+1,\ldots,n$

    $x_i = x_i \Big/ (\alpha_i - \alpha_{i-k-1})$

  For $i = k,\ldots,n-1$

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

1.2.Desarrollo de Procedimientos

En esta parte se desarrolla manualmente el algoritmo Dual con la finalidad de poder observar con claridad la manipulación de los subíndices dentro del proceso iterativo.

A continuación $c = U^T f$, esto equivale al Paso No. 1 del algoritmo Dual...

Recordando:

$$c^{(0)} = f$$

$$For\ k = 0,1,\ldots,n-1$$

$$For\ j = n, n-1, \ldots, k+1$$

$$c_j^{(k+1)} = \left(c_j^{(k)} - c_{j-1}^{(k)}\right) \Big/ \left(\alpha_j - \alpha_{j-k-1}\right)$$

Desarrollando:

$k = 0$;

$j = n$; hasta $k+1$

$$c_n^{(0)} = \left(c_n^{(0)} - c_{n-1}^{(0)}\right)/\left(\alpha_n - \alpha_{n-1}\right)$$

$$c_{n-1}^{(0)} = \left(c_{n-1}^{(0)} - c_{n-2}^{(0)}\right)/\left(\alpha_{n-1} - \alpha_{n-2}\right)$$

$$\vdots \qquad\qquad \vdots$$

$$c_2^{(0)} = \left(c_2^{(0)} - c_1^{(0)}\right)/\left(\alpha_2 - \alpha_1\right)$$

$$c_1^{(0)} = \left(c_1^{(0)} - c_0^{(0)}\right)/\left(\alpha_1 - \alpha_0\right)$$

$k = 1$;

$j = n$; hasta $k+1$

$$c_n^{(1)} = \left(c_n^{(1)} - c_{n-1}^{(1)}\right)/\left(\alpha_n - \alpha_{n-2}\right)$$

$$c_{n-1}^{(1)} = \left(c_{n-1}^{(1)} - c_{n-2}^{(1)}\right)/\left(\alpha_{n-1} - \alpha_{n-3}\right)$$

$$\vdots \qquad\qquad \vdots$$

$$c_3^{(1)} = \left(c_3^{(1)} - c_2^{(1)}\right)/\left(\alpha_3 - \alpha_1\right)$$

$$c_2^{(1)} = \left(c_2^{(1)} - c_1^{(1)}\right)/\left(\alpha_2 - \alpha_0\right)$$

$k = 2$;

$j = n$; hasta $k + 1$

$$c_n^{(2)} = \left(c_n^{(2)} - c_{n-1}^{(2)}\right)/(\alpha_n - \alpha_{n-3})$$

$$c_{n-1}^{(2)} = \left(c_{n-1}^{(2)} - c_{n-2}^{(2)}\right)/(\alpha_{n-1} - \alpha_{n-4})$$

$$\vdots \qquad\qquad \vdots$$

$$c_4^{(2)} = \left(c_4^{(2)} - c_3^{(2)}\right)/(\alpha_4 - \alpha_1)$$

$$c_3^{(2)} = \left(c_3^{(2)} - c_2^{(2)}\right)/(\alpha_3 - \alpha_0)$$

$k = 3$;

$j = n$; hasta $k + 1$

$$c_n^{(3)} = \left(c_n^{(3)} - c_{n-1}^{(3)}\right)/(\alpha_n - \alpha_1)$$

$$c_{n-1}^{(3)} = \left(c_{n-1}^{(3)} - c_{n-2}^{(3)}\right)/(\alpha_{n-1} - \alpha_0)$$

$k = n - 1$

$$c_n^{(n-1)} = \left(c_n^{(n-1)} - c_{n-1}^{(n-1)}\right)/(\alpha_n - \alpha_0)$$

Ahora haciendo el procedimiento anterior en el lenguaje de vectores y matrices queda $\mathrm{c} = U^{\mathrm{T}}\mathrm{f}$ y se observa así:

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{pmatrix} = D_{n-1}^{-1} M_{n-1} \cdots D_0^{-1} M_0 \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{n-1} \end{pmatrix} \qquad (**)$$

con la matriz diagonal D_k y $M_k = L_k(1)$, que a continuacón se presentan...

$$D_{n-1}^{-1} = \begin{bmatrix} 1 & & & & 0 \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \vdots \\ 0 & & & \cdots & (\alpha_n - \alpha_0)^{-1} \end{bmatrix} \quad ; \quad M_{n-1} = \begin{bmatrix} 1 & & & & 0 \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \vdots \\ 0 & & & -1 & 1 \end{bmatrix}$$

$$D_{n-2}^{-1}=\begin{bmatrix}1 & & & & 0\\ & 1 & & & \\ & & \ddots & & \vdots\\ & & & (\alpha_{n-1}-\alpha_0)^{-1} & 0\\ 0 & & \cdots & 0 & (\alpha_n-\alpha_1)^{-1}\end{bmatrix};\ M_{n-2}=\begin{bmatrix}1 & & & & 0\\ & \ddots & & & \\ & & 1 & & \\ & & -1 & 1 & \vdots\\ 0 & & \cdots & -1 & 1\end{bmatrix}$$

$$D_{n-3}^{-1}=\begin{bmatrix}1 & 0 & 0 & 0 & 0\\ 0 & \ddots & \vdots & \vdots & \\ 0 & \cdots & (\alpha_{n-2}-\alpha_0)^{-1} & 0 & \vdots\\ 0 & \cdots & 0 & (\alpha_{n-1}-\alpha_1)^{-1} & 0\\ 0 & \cdots & 0 & 0 & (\alpha_n-\alpha_2)^{-1}\end{bmatrix};M_{n-3}=\begin{bmatrix}1 & & & & 0\\ & 1 & & & \\ & -1 & 1 & & \\ & & -1 & 1 & \vdots\\ 0 & & \cdots & -1 & 1\end{bmatrix}$$

$$\vdots \qquad\qquad \vdots$$

$$D_0^{-1}=\begin{bmatrix}1 & 0 & 0 & 0 & 0\\ 0 & (\alpha_1-\alpha_0)^{-1} & \vdots & \vdots & \\ 0 & \cdots & (\alpha_2-\alpha_1)^{-1} & 0 & \vdots\\ 0 & \cdots & 0 & \ddots & 0\\ 0 & \cdots & 0 & 0 & (\alpha_n-\alpha_{n-1})^{-1}\end{bmatrix};\ M_0=\begin{bmatrix}1 & & & & 0\\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & -1 & 1 & \vdots\\ 0 & & \cdots & -1 & 1\end{bmatrix}$$

Efectuando el producto (**)...

$$\begin{bmatrix}1 & & & & 0\\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & -1 & 1 & \vdots\\ 0 & & \cdots & -1 & 1\end{bmatrix}\begin{pmatrix}f_0\\ f_1\\ f_2\\ \vdots\\ f_n\end{pmatrix}=\begin{pmatrix}f_0\\ f_1-f_0\\ f_2-f_1\\ \vdots\\ f_n-f_{n-1}\end{pmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & (\alpha_1-\alpha_0)^{-1} & \vdots & \vdots & \\ 0 & \cdots & (\alpha_2-\alpha_1)^{-1} & 0 & \vdots \\ 0 & \cdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 0 & (\alpha_n-\alpha_{n-1})^{-1} \end{bmatrix} \begin{pmatrix} f_0 \\ f_1-f_0 \\ f_2-f_1 \\ \vdots \\ f_n-f_{n-1} \end{pmatrix} = \begin{pmatrix} f_0 \\ (f_1-f_0)/(\alpha_1-\alpha_0) \\ (f_2-f_1)/(\alpha_2-\alpha_1) \\ \vdots \\ (f_n-f_{n-1})/(\alpha_n-\alpha_{n-1}) \end{pmatrix}$$

Hasta aquí el producto que se ha efectuado es $\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} = \cdots D_0^{-1} M_0 \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix}$, pero con el objeto de no escribir demasiado, ya que el procedimiento es similar en su parte intermedia, llegamos a la parte final :

$$\begin{bmatrix} 1 & & & 0 \\ & 1 & & \\ & & 1 & \\ & & \ddots & \vdots \\ 0 & & \cdots & (\alpha_n-\alpha_0)^{-1} \end{bmatrix} \begin{bmatrix} 1 & & & & 0 \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \vdots \\ 0 & & \cdots & -1 & 1 \end{bmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ (f_n-f_{n-1})/(\alpha_n-\alpha_{n-1}) \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$$

Por tanto, se observa a través de los procedimientos que la Parte No.1 del algoritmo Dual equivale a $\mathrm{c} = U^{\mathrm{T}}\mathrm{f}$.

Se han obtenido las diferencias divididas del algoritmo de interpolación polinomial de Newton mediante el lenguaje de vectores y matrices. Y ya que se obtienen las diferencias divididas, proseguimos a obtener los coeficientes a's.

En seguida se analiza la Parte No.2 de ese mismo algoritmo Dual, y que equivale a $\mathrm{a} = L^{T}\mathrm{c}$

Recordando: Parte No. 2 (Horner) Dual No-confluente

$a^{(n)} = c$

For $k = n-1, \ldots 1, 0$

For $j = k, k+1, \ldots, n-1$

$$a_j^{(k)} = a_j^{(k)} - \alpha_k a_{j+1}^{(k+1)}$$

Desarrollando:

$k = n-1$; hasta 0

$j = k$, $k+1$, hasta $n-1$

$$a_{n-1}^{(n-1)} = a_{n-1}^{(n-1)} - \alpha_{n-1} a_n^{(n)}$$

$k = n-2$

$$a_{n-2}^{(n-2)} = a_{n-2}^{(n-2)} - \alpha_{n-2} a_{n-1}^{(n-1)}$$
$$a_{n-1}^{(n-2)} = a_{n-1}^{(n-2)} - \alpha_{n-2} a_n^{(n-1)}$$

$k = n-3$

$$a_{n-3}^{(n-3)} = a_{n-3}^{(n-3)} - \alpha_{n-3} a_{n-2}^{(n-2)}$$
$$a_{n-2}^{(n-3)} = a_{n-2}^{(n-3)} - \alpha_{n-3} a_{n-1}^{(n-2)}$$
$$a_{n-1}^{n-3} = a_{n-1}^{(n-3)} - \alpha_{n-3} a_n^{(n-2)}$$

$k = n-4$

$$a_{n-4}^{(n-4)} = a_{n-4}^{(n-4)} - \alpha_{n-4} a_{n-3}^{(n-3)}$$
$$a_{n-3}^{(n-4)} = a_{n-3}^{(n-4)} - \alpha_{n-4} a_{n-2}^{(n-3)}$$
$$a_{n-2}^{(n-4)} = a_{n-2}^{(n-4)} - \alpha_{n-4} a_{n-1}^{(n-3)}$$
$$a_{n-1}^{(n-4)} = a_{n-1}^{(n-4)} - \alpha_{n-4} a_n^{(n-3)}$$

$\vdots$

$\vdots$

$k = n-n = 0$

$$a_0^{(0)} = a_0^{(0)} - \alpha_0 a_1^{(1)}$$
$$a_1^{(0)} = a_1^{(0)} - \alpha_0 a_2^{(1)}$$
$$a_2^{(0)} = a_2^{(0)} - \alpha_0 a_3^{(1)}$$
$$a_3^{(0)} = a_3^{(0)} - \alpha_0 a_4^{(1)}$$
$$\vdots$$
$$a_{n-1}^{(0)} = a_{n-1}^{(0)} - \alpha_0 a_n^{(1)}$$

Con ésta última iteración se terminan de obtener los coeficientes a's. Ahora mediante la operación de vectores y matrices, podemos llegar también a ello mediante $\mathrm{a} = L^T \mathrm{c}$, donde L es la matriz triangular inferior definida por

$$L^T = N_0^T N_1^T \cdots N_{n-1}^T. \text{ donde } N_k = L_k(\alpha_k).$$

Aplicando la regla de la matriz bidiagonal inferior...

$$N_0 = \begin{bmatrix} 1 & & & & 0 \\ -\alpha_0 & 1 & & & \\ & -\alpha_0 & 1 & & \\ & & \ddots & \ddots & \vdots \\ 0 & & \cdots & -\alpha_0 & 1 \end{bmatrix} \qquad N_0^T = \begin{bmatrix} 1 & -\alpha_0 & & & 0 \\ 0 & 1 & -\alpha_0 & & \\ & 0 & 1 & \ddots & \\ & & \ddots & \ddots & -\alpha_0 \\ 0 & & \cdots & 0 & 1 \end{bmatrix}$$

$$N_1 = \left[\begin{array}{c|ccc} 1 & & & 0 \\ \hline & 1 & & \\ & -\alpha_1 & 1 & \\ & & -\alpha_1 & \ddots \\ 0 & & & \ddots \; 1 \end{array}\right] \qquad N_1^T = \begin{bmatrix} 1 & & & 0 \\ & 1 & -\alpha_1 & \\ & & 1 & -\alpha_1 \\ & & & \ddots \; -\alpha_1 \\ 0 & & & 1 \end{bmatrix}$$

$$N_2 = \left[\begin{array}{c|ccc} 1 & & & 0 \\ & 1 & & \\ \hline & 1 & & \\ & -\alpha_2 & \ddots & \\ 0 & & \ddots & 1 \end{array}\right] \qquad N_2^T = \begin{bmatrix} 1 & & & 0 \\ & 1 & & \\ & & 1 & -\alpha_1 \\ & & & \ddots \; -\alpha_1 \\ 0 & & & 1 \end{bmatrix}$$

$$\vdots$$

Y así sucesivamente. Ahorrándonos los pasos intermedios hacemos el último :

$$N_{n-1} = \left[\begin{array}{ccc|c} 1 & & & 0 \\ & 1 & & \\ & & \ddots & \\ \hline & & & 1 \\ 0 & & & -\alpha_{n-1} \end{array}\right] \qquad N_{n-1}^T = \left[\begin{array}{cccc} 1 & & & 0 \\ & 1 & & \\ & & 1 & \\ & & \ddots & -\alpha_{n-1} \\ 0 & & & 1 \end{array}\right]$$

Evaluando el producto $\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = N_0^T N_1^T \cdots N_{n-1}^T \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$, tenemos lo siguiente

$$\begin{bmatrix} 1 & & & & 0 \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & -\alpha_{n-1} \\ 0 & & & & 1 \end{bmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} - \alpha_{n-1} c_n \\ c_n \end{pmatrix}$$

Vamos al siguiente producto $\cdots N_{n-2}^T N_{n-1}^T \mathrm{c}$

$$\begin{bmatrix} 1 & & & & 0 \\ & \ddots & & & \\ & & 1 & \alpha_{n-2} & \\ & & & 1 & -\alpha_{n-2} \\ 0 & & & & 1 \end{bmatrix} \begin{pmatrix} c_0 \\ \vdots \\ c_{n-2} \\ c_{n-1} - \alpha_{n-1} c_n \\ c_n \end{pmatrix} = \begin{pmatrix} c_0 \\ \vdots \\ c_{n-2} - \alpha_{n-2}\left(c_{n-1} - \alpha_{n-1} c_n\right) \\ c_{n-1} - \alpha_{n-1} c_n - \alpha_{n-2} c_n \\ c_n \end{pmatrix}$$

Se observa que

$$c_{n-1} = c_{n-1} - \alpha_{n-1} c_n$$

equivale a la parte No. 2 (Horner), cuando $k = n-1$, veamoslo :

$$a_{n-1}^{(n-1)} = a_{n-1}^{(n-1)} - \alpha_{n-1} a_n^{(n)}$$

además

$$c_{n-2} = c_{n-2} - \alpha_{n-2}c_{n-1} = c_{n-2} - \alpha_{n-2}\left(c_{n-1} - \alpha_{n-1}c_n\right)$$

$$c_{n-1} = c_{n-1} - \alpha_{n-1}c_n - \alpha_{n-2}c_n$$

Y estos dos últimos renglones equivalen al algoritmo en la Parte No.2 cuando $k = n-2$, veamos...

$$a_{n-2}^{(n-2)} = a_{n-2}^{(n-1)} - \alpha_{n-2}a_{n-1}^{(n-1)}$$

Ejecutando la siguiente iteración se llega...

$$a_{n-1}^{(n-2)} = a_{n-1}^{(n-1)} - \alpha_{n-2}a_n^{(n-1)}$$

Y van efectuandose sucesivamente los productos $N_0^T N_1^T \cdots N_{n-1}^T \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$ lo

que equivale a la expresión $\mathrm{a} = L^T\mathrm{c}$

Veamos el producto final :

$$\begin{bmatrix} 1 & -\alpha_0 & & & 0 \\ & \ddots & & & \\ & & 1 & -\alpha_0 & \\ & & & 1 & -\alpha_0 \\ 0 & & & & 1 \end{bmatrix} \begin{pmatrix} c_0 \\ \vdots \\ c_{n-2} \\ c_{n-1} \\ c_n \end{pmatrix} = \begin{pmatrix} c_0 - \alpha_0 c_1 \\ \vdots \\ c_{n-2} - \alpha_0 c_{n-1} \\ c_{n-1} - \alpha_0 c_n \\ c_n \end{pmatrix}$$

con lo cual hemos finalizado

$$c_0 = c_0 - \alpha_0 c_1$$

$$c_1 = c_1 - \alpha_0 c_2$$

$$c_2 = c_2 - \alpha_0 c_3$$

$$\vdots$$

$$c_{n-2} = c_{n-2} - \alpha_0 c_{n-1}$$

$$c_{n-1} = c_{n-1} - \alpha_0 c_n$$

lo que es equivalente al algoritmo en la Parte No.2 (Horner) cuando $k = 0$

$$a_0^{(0)} = a_0^{(0)} - \alpha_0 a_1^{(1)}$$

$$a_1^{(0)} = a_1^{(0)} - \alpha_0 a_2^{(1)}$$

$$
\begin{aligned}
a_2^{(0)} &= a_2^{(0)} - \alpha_0 a_3^{(1)} \\
a_3^{(0)} &= a_3^{(0)} - \alpha_0 a_4^{(1)} \\
&\vdots \\
a_{n-1}^{(0)} &= a_{n-1}^{(0)} - \alpha_0 a_n^{(1)}
\end{aligned}
$$

Por lo tanto a = L^Tc proporciona la solución a la Parte No. 2 del algoritmo Dual.

Consecuentemente una solución para el sistema de Vandermonde Vx = b; es:

$$\mathrm{x} = V^{-1}\mathrm{b} = \left(M_0^T D_0^{-1} \cdots M_{n-1}^T D_{n-1}^{-1}\right)\left(N_{N-1} \cdots N_1 N_0\right)b$$

Observamos que $V^{-1} = UL$.

Esto en leguaje de vectores y matrices es lo que anteriormente se llamó algoritmo Primal.

1.3. Algorimos Progresivos
1.3.1. Dual

Aquí consideramos tanto para el Dual ($V^T\mathrm{a} = \mathrm{f}$) como para el Primal ($V\mathrm{x} = \mathrm{b}$), el hecho de que se agregue un nuevo valor α. Esto, con la finalidad de actualizar el sistema sin necesidad de efectuar nuevamente los cálculos desde el inicio. La representación nueva para ambos sistemas es

$$V_n^T \mathrm{a}_n = \mathrm{f}_n \quad \text{y} \quad V_n \mathrm{x}_n = \mathrm{b}_n$$

donde la matriz de Vandermonde ahora se indica como $V_n = V(\alpha_0, \alpha_1, \ldots, \alpha_n)$

El Paso No.1 del algoritmo Dual, que consiste en encontrar las diferencias divididas, en el lenguaje matricial equivale a $\mathrm{c} = U^{\mathrm{T}}\mathrm{f}$. Entonces cuando se agrega un nuevo valor α_n, ésta parte queda en lenguaje algebraico como:

$$\mathrm{c}^{(n)} = U_n^T \mathrm{f}_n$$

Entonces cuando el nuevo valor es agregado, los primeros $n-1$ valores no sufren alteración en $\mathrm{c}^{(n)}$, y lo único que tendrá que calcularse es el valor $c_n^{(n)}$. Almacenamos las cantidades anteriores, es decir $c_{n-1}^{(k)}$ para $k = 0, 1, \ldots, n-1$.

$c^{(0)} = f$	valores calculados						
$c_{n-1}^{(k=0)}$	$c_{n-1}^{(k=1)}$	$c_{n-1}^{(k=2)}$	$c_{n-1}^{(k=3)}$	$c_{n-1}^{(k=4)}$	$c_{n-1}^{(k=5)}$	...	$c_{n-1}^{(k=n-1)}$
$c_{n-2}^{(k=0)}$	$c_{n-2}^{(k=1)}$	$c_{n-2}^{(k=2)}$	$c_{n-2}^{(k=3)}$	...	$c_{n-2}^{(k=n-2)}$		
$c_{n-3}^{(k=0)}$	$c_{n-3}^{(k=1)}$	$c_{n-3}^{(k=2)}$	...	$c_{n-3}^{(k=n-3)}$			
⋮	⋮						
⋮							
$c_1^{(k=0)}$	$c_1^{(k=1)}$						
$c_0^{(k=0)}$							

Si se almacenan los $n-1$ valores anteriores para $c_{n-1}^{(k)}$, $k = 0, 1, \ldots, n-1$, en este caso se han guardado $c_{n-1}^{(k=0)}, c_{n-1}^{(k=1)}, c_{n-1}^{(k=2)}, c_{n-1}^{(k=3)}, \ldots, c_{n-1}^{(k=n-1)}$. Entonces haciendo $j = n$

en el Paso No. 1 del algoritmo Dual, lo único que resta es calcular los valores que identificamos por $c_n^{(k)}$, $k = 0,1,...,n$.

Esto es en términos de la interpolación polinomial; en términos de álgebra matricial dijimos que los $\mathrm{c}^{(k)}$ son recursivamente generados por

$$\mathrm{c}^{(0)} = \mathrm{f}, \qquad \mathrm{c}^{(k+1)} = D_k^{-1} M_k \mathrm{c}^{(k)}, \qquad M_k = L_k(1), \qquad k = 0,1,...,n-1$$

Nuevamente, si los primeros $\mathrm{c}^{(n-1)}$ permanecen sin cambio, calculamos entonces $\mathrm{c}^{(n)}$, para cuando agreguemos un nuevo valor α_n.

Hasta aquí utilizamos la Parte No. 1 del Dual, o lo que es lo mismo aplicamos el concepto de las diferencias divididas a los progresivos. Ahora se emplea la Parte No.2 para obtener los coeficientes a_i. Pero antes hacemos la observación que si se agrega un nuevo α_n se agrega un vector al producto de matrices anteriormente mencionado. Si se dijo $V^{-1} = UL$, se supone que se calcularon hasta los $n-1$ coeficientes, o sea $V_{n-1}^{-1} = U_{n-1} L_{n-1}$. Agregando el nuevo valor α_n, se ve que $V_n^{-1} = U_n L_n$. Visto de otra manera, (en forma particionada)

$$\left[\begin{array}{ccc|c} & V_{n-1} & & \begin{matrix}1\\ \cdots\\ x_n^{n-1}\end{matrix} \\ \hline x_0^n & \cdots & x_{n-1}^n & x_n^n \end{array}\right]^{-1} = \left[\begin{array}{ccc|c} & U_{n-1} & & \begin{matrix}u_0^{(n)}\\ \cdots\\ u_{n-1}^{(n)}\end{matrix} \\ \hline 0 & & & u_n^{(n)} \end{array}\right] \left[\begin{array}{ccc|c} & L_{n-1} & & 0 \\ \hline m_0^{(n)} & \cdots & m_{n-1}^{(n)} & m_n^{(n)} \end{array}\right]$$

Regresando a la Parte No. 2 tenemos

$$\mathrm{a}_n = L_n^T c^{(n)} = \begin{pmatrix} \mathrm{a}_{n-1} \\ 0 \end{pmatrix} + c_n^{(n)} \left(\mathrm{m}^{(n)}\right)^T$$

donde $\mathrm{m}^{(n)}$ es el último renglón en la matriz L_n. Para llegar a

$$P_n(x) = m_0^{(n)} + m_1^{(n)} x + \cdots + m_n^{(n)} x^n$$

observamos que $L_n^T = \left[\begin{array}{c|c} L_{n-1}^T & \begin{matrix} m_0^{(n)} \\ \vdots \\ m_{n-1}^{(n)} \end{matrix} \\ \hline 0 & m_n^{(n)} \end{array}\right]$ se compone de $L^T = N_0^T N_1^T \cdots N_{n-1}^T$,

donde $N_k = L_k(\alpha_k)$ para $k = n-1,\ldots,1,0$. Veamos esta parte posteriormente, en el desarrollo de procedimientos progresivos.

Por otro lado, observamos el método de Newton, y recordamos cómo se obtienen los polinomios

$$P_0(x) = 1 \qquad \text{y} \qquad P_k(x) = \prod_{i=0}^{k-1}(x-\alpha_i) \quad \text{para} \quad k = 1,2,\ldots n$$

$$P_0(x) = 1$$

$$P_1(x) = (x-\alpha_0)$$

$$P_2(x) = (x-\alpha_0)(x-\alpha_1)\,x^2 - (\alpha_0+\alpha_1)x + \alpha_0\alpha_1$$

$$P_3(x) = (x-\alpha_0)(x-\alpha_1)(x-\alpha_2) = (x-\alpha_0)P_2(x)$$

$$= (x-\alpha_0)\left[(x-\alpha_1)(x-\alpha_2)\right]$$

$$= (x-\alpha_0)\left[x^2 - (\alpha_1+\alpha_2)x + \alpha_1\alpha_2\right]$$

$$= x^3 - (\alpha_1+\alpha_2)x^2 + \alpha_1\alpha_2 x - \alpha_0 x^2 + (\alpha_0\alpha_1 + \alpha_0\alpha_2)x - \alpha_0\alpha_1\alpha_2$$

reagrupando términos $= -\alpha_0\alpha_1\alpha_2 + (\alpha_0\alpha_1 + \alpha_0\alpha_2 + \alpha_1\alpha_2)x - (\alpha_0+\alpha_1+\alpha_2)x^2 + x^3$.

Para el caso general definimos a

$$P_{n-1}(x) = m_0^{(n-1)} + m_1^{(n-1)}x + \cdots + m_{n-2}^{(n-1)}x^{n-2} + m_{n-1}^{(n-1)}x^{n-1}$$

entonces al evaluar el polinomio $P_n(x) = (x-\alpha_{n-1})P_{n-1}(x)$, se sustituye $P_{n-1}(x)$ y da

$$P_n(x) = (x-\alpha_{n-1})\left[\; m_0^{(n-1)} + m_1^{(n-1)}x + \cdots + m_{n-2}^{(n-1)}x^{n-2} + m_{n-1}^{(n-1)}x^{n-1} \;\right]$$

$$= m_0^{(n-1)}x + m_1^{(n-1)}x^2 + \cdots + m_{n-2}^{(n-1)}x^{n-1} + m_{n-1}^{(n-1)}x^n$$

$$- \alpha_{n-1}m_0^{(n-1)} - \alpha_{n-1}m_1^{(n-1)}x \cdots - \alpha_{n-1}m_{n-2}^{(n-1)}x^{n-2} - \alpha_{n-1}m_{n-1}^{(n-1)}x^{n-1}$$

igualando y agrupando términos iguales

$$
\begin{aligned}
m_0^{(n)} &= -\alpha_{n-1} m_0^{(n-1)} \\
m_0^{(n)} &= m_0^{(n-1)} - \alpha_{n-1} m_1^{(n-1)} \\
&\vdots \qquad\qquad \vdots \\
m_{n-1}^{(n)} &= m_{n-2}^{(n-1)} - \alpha_{n-1} m_{n-1}^{(n-1)} \\
m_n^{(n)} &= m_{n-1}^{(n-1)}
\end{aligned}
$$

Además si observamos la matriz diagonal L_n^T en su última columna, la cual es $m^{(n)}$, donde por ejemplo para una $n = 3$,

$$
\begin{aligned}
m_0^{(3)} &= -\alpha_0 \alpha_1 \alpha_2 \\
m_1^{(3)} &= \alpha_0 \alpha_1 + \alpha_1 \alpha_2 + \alpha_0 \alpha_2 \\
m_2^{(3)} &= -\alpha_0 - \alpha_1 - \alpha_2 \\
m_3^{(3)} &= 1, \text{ etc.}
\end{aligned}
$$

Estos números no son otra cosa que las entradas de L_n^T. También son los coeficientes del polinomio.

$$P_n(z) = m_0^{(n)} + m_1^{(n)} z + \cdots + m_n^{(n)} z^n$$

De esta manera se elabora el algoritmo para obtener $m^{(n)}$:

Algoritmo Progresivo para el sistema Dual

$$
\begin{aligned}
& a_0^{(0)} = c_0^{(0)} = f_0 \; ; \; m_0^{(0)} = 1 \\
\text{For } n = 1,2,3,\cdots \quad & c_n^{(0)} = f_n \\
& c_n^{(k+1)} = \left(c_n^{(k)} - c_{n-1}^{(k)} \right) / \left(\alpha_n - \alpha_{n-k-1} \right); \; k = 0,1,\ldots,n-1 \\
& m_n^{(n)} = 1 \\
& a_n^{(n)} = c_n^{(n)} \\
& m_{-1}^{(n-1)} = 0 \\
& m_k^{(n)} = m_{k-1}^{(n-1)} - \alpha_{n-1} m_k^{(n-1)} \; ; \; k = n-1,\ldots,1,0 \\
& a_k^{(n)} = a_k^{(n-1)} + m_k^{(n)} c_n^{(n)} \; ; \; k = n-1,\ldots,1,0
\end{aligned}
$$

1.3. Algoritmos Progresivos
1.3.2. Primal

El sistema $V\mathrm{x}=\mathrm{b}$ al agregarse un nuevo valor α, puede verse como $V_n\mathrm{x}_n=\mathrm{b}_n$ donde $V_n=V(\alpha_0,\alpha_1,\ldots,\alpha_n)$. A continuación lo desarrollamos con la finalidad de ver que no es necesario volver a computar las operaciones anteriores.

Entonces tenemos que en la Parte No. 1 del algoritmo en el lenguaje del álgebra de matrices se expresa como :

$$\mathrm{d}^{(n)}=L_n\mathrm{b}_n$$

Visto matricialmente

$$\begin{bmatrix} d^{(0)} \\ d^{(1)} \\ d^{(2)} \\ \vdots \\ d^{(n-1)} \\ d^{(n)} \end{bmatrix} = N_nN_{n-1}\cdots N_1N_0 \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}$$

Una vez más, los primeros $n-1$ componentes permanecen sin cambio alguno, los $d_{n-1}^{(k)}$; $k=0,1,\cdots,n-1$, han sido guardados ó almacenados (en este caso los elementos que se guardan son $d_{n-1}^{(0)},d_{n-1}^{(1)},d_{n-1}^{(2)},\ldots d_{n-1}^{(n-1)}$) y ahora lo único que se tiene que calcular son los $d_n^{(k)}$; $k=0,1,\cdots,n$,

Recordemos que $L=N_{n-1}N_{n-2}\cdots N_1N_0$ donde $N_k=L_k(\alpha_k)$ y $L_k(\alpha)$ es la bidiagonal anteriormente definida y es una matriz de tamaño $(n+1)$ x $(n+1)$. A continuación mostramos la matriz L para $n=5$, solo que por motivos de tamaño en la presentación quedó impresa en la siguiente hoja.

Esta matriz L para una $n=5$ se presenta así por motivos de legibilidad

1	0	0	0	0	0
$-\alpha_0$	1	0	0	0	0
$\alpha_0\alpha_1$	$-\alpha_0-\alpha_1$	1	0	0	0
$-\alpha_0\alpha_1\alpha_2$	$\alpha_0\alpha_1+\alpha_0\alpha_2+\alpha_1\alpha_2$	$-\alpha_0-\alpha_1-\alpha_2$	1	0	0
$\alpha_0\alpha_1\alpha_2\alpha_3$	$-\alpha_0\alpha_1\alpha_2$ $-\alpha_0\alpha_2\alpha_3-\alpha_1\alpha_2\alpha_3$ $-\alpha_0\alpha_1\alpha_3$	$\alpha_0\alpha_1+\alpha_0\alpha_2+\alpha_1\alpha_2$ + $\alpha_0\alpha_3+\alpha_1\alpha_3+\alpha_2\alpha_3$	$-\alpha_0-\alpha_1-\alpha_2-\alpha_3$	1	0
$-\alpha_0\alpha_1\alpha_2\alpha_3\alpha_4$	$\alpha_0\alpha_1\alpha_2\alpha_3+$ $\alpha_0\alpha_2\alpha_3\alpha_4+$ $\alpha_1\alpha_2\alpha_3\alpha_4+$ $\alpha_0\alpha_1\alpha_3\alpha_4+$ $\alpha_0\alpha_1\alpha_2\alpha_4$	$-\alpha_0\alpha_1\alpha_2$ $-\alpha_0\alpha_2\alpha_3-\alpha_1\alpha_2\alpha_3$ $-\alpha_0\alpha_1\alpha_3$ $-\alpha_0\alpha_2\alpha_4-\alpha_1\alpha_2\alpha_4$ $-\alpha_0\alpha_1\alpha_4-\alpha_0\alpha_3\alpha_4$ $-\alpha_1\alpha_3\alpha_4-\alpha_2\alpha_3\alpha_4$	$\alpha_0\alpha_1+\alpha_0\alpha_2+\alpha_1\alpha_2$ + $\alpha_0\alpha_3+\alpha_1\alpha_3+\alpha_2\alpha_3$ + $\alpha_0\alpha_4+\alpha_1\alpha_4+\alpha_2\alpha_4$ + $\alpha_3\alpha_4$	$-\alpha_0-\alpha_1-\alpha_2-\alpha_3$ $-\alpha_4$	1

Desarrollando para encontrar las $d^{(k)}$:

$$d^{(0)} = b_0$$

$$d^{(1)} = b_0(-\alpha_0) + b_1$$

$$d^{(2)} = b_0(\alpha_0\alpha_1) + b_1(-\alpha_0 - \alpha_1) + b_2$$

$$d^{(3)} = b_0(-\alpha_0\alpha_1\alpha_2) + b_1(\alpha_0\alpha_2 + \alpha_1\alpha_2 + \alpha_0\alpha_1) + b_2(-\alpha_0 - \alpha_1 - \alpha_2) + b_3$$

$$d^{(4)} = b_0(\alpha_0\alpha_1\alpha_2\alpha_3) + b_1(-\alpha_0\alpha_1\alpha_2 - \alpha_0\alpha_2\alpha_3 - \alpha_1\alpha_2\alpha_3 - \alpha_0\alpha_1\alpha_3) +$$
$$+ b_2(\alpha_0\alpha_2 + \alpha_1\alpha_2 + \alpha_0\alpha_1 + \alpha_0\alpha_3 + \alpha_1\alpha_3 + \alpha_2\alpha_3) + b_3(-\alpha_0 - \alpha_1 - \alpha_2 - \alpha_3) + b_4$$

$$d^{(5)} = b_0(-\alpha_0\alpha_1\alpha_2\alpha_3\alpha_4) + b_1(\alpha_0\alpha_1\alpha_2\alpha_3 + \alpha_0\alpha_2\alpha_3\alpha_4 + \alpha_1\alpha_2\alpha_3\alpha_4 + \alpha_0\alpha_1\alpha_3\alpha_4 + \alpha_0\alpha_1\alpha_2\alpha_4) +$$
$$+ b_2(-\alpha_0\alpha_1\alpha_2 - \alpha_0\alpha_2\alpha_3 - \alpha_1\alpha_2\alpha_3 - \alpha_0\alpha_1\alpha_3 - \alpha_0\alpha_2\alpha_4 - \alpha_1\alpha_2\alpha_4 - \alpha_0\alpha_1\alpha_4$$
$$- \alpha_0\alpha_3\alpha_4 - \alpha_1\alpha_3\alpha_4 - \alpha_2\alpha_3\alpha_4)$$
$$+ b_3(\alpha_0\alpha_2 + \alpha_1\alpha_2 + \alpha_0\alpha_1 + \alpha_0\alpha_3 + \alpha_1\alpha_3 + \alpha_2\alpha_3 + \alpha_0\alpha_4 + \alpha_1\alpha_4 + \alpha_2\alpha_4 + \alpha_3\alpha_4) +$$
$$+ b_4(-\alpha_0 - \alpha_1 - \alpha_2 - \alpha_3 - \alpha_4) + b_5 .$$

$$\vdots$$

$$d^{(n)} = b_0(\ldots) + b_1(\ldots) + b_2(\ldots) + b_3(\ldots) + b_4(\ldots) + b_5(\ldots) + b_6(\ldots) + \ldots b_n$$

Para la parte No. 2 tenemos $x_n = U_n d^{(n)}$, donde la

$$U_n = \left[\begin{array}{c|c} & u_0^{(n)} \\ U_{n-1} & \cdots \\ & u_{n-1}^{(n)} \\ \hline 0 & u_n^{(n)} \end{array}\right]$$

$$U_{n-1} = M_0^T D_0^{-1} \cdots M_{n-1}^T D_{n-1}^{-1}; \qquad M_k = L_k(1) \qquad \text{y}$$

$$D_k = \text{diag}\left(\underbrace{1,\ldots,1}_{k+1}, (\alpha_{k+1} - \alpha_0), \ldots, (\alpha_n - \alpha_{n-k-1})\right)$$

Desarrollamos el producto de matrices para obtener $U_{n-1} = M_0^T D_0^{-1} \cdots M_{n-1}^T D_{n-1}^{-1}$. Recordemos que U_{n-1} es una matriz triangular superior. Para una $n = 4$, por motivos de presentación queda ilustrada esta matriz en la siguiente página.

1	$-(\alpha_1-\alpha_0)^{-1}$	$(\alpha_2-\alpha_0)^{-1}(\alpha_1-\alpha_0)^{-1}$	$-(\alpha_3-\alpha_0)^{-1}(\alpha_2-\alpha_0)^{-1}(\alpha_1-\alpha_0)^{-1}$	$-(\alpha_1-\alpha_0)^{-1}\left[-(\alpha_4-\alpha_0)^{-1}(\alpha_3-\alpha_0)^{-1}(\alpha_2-\alpha_0)^{-1}\right]$
0	$(\alpha_1-\alpha_0)^{-1}$	$-(\alpha_2-\alpha_0)^{-1}(\alpha_1-\alpha_0)^{-1}$ $-(\alpha_2-\alpha_0)^{-1}(\alpha_2-\alpha_1)^{-1}$	$(\alpha_3-\alpha_0)^{-1}(\alpha_2-\alpha_0)^{-1}(\alpha_1-\alpha_0)^{-1}$ $-(\alpha_2-\alpha_0)^{-1}\left[-(\alpha_3-\alpha_0)^{-1}(\alpha_2-\alpha_0)^{-1}-\right.$ $\left.(\alpha_3-\alpha_0)^{-1}(\alpha_3-\alpha_1)^{-1}\right]$	$-(\alpha_1-\alpha_0)^{-1}\left[-(\alpha_4-\alpha_0)^{-1}(\alpha_3-\alpha_0)^{-1}(\alpha_2-\alpha_0)^{-1}\right]$ $-(\alpha_2-\alpha_1)^{-1}\left\{(\alpha_4-\alpha_0)^{-1}(\alpha_3-\alpha_0)^{-1}(\alpha_2-\alpha_0)^{-1}-\right.$ $(\alpha_3-\alpha_1)^{-1}\left[-(\alpha_4-\alpha_0)^{-1}(\alpha_3-\alpha_0)^{-1}-(\alpha_4-\alpha_0)^{-1}\right.$ $\left.\left.(\alpha_4-\alpha_1)^{-1}\right]\right\}$
0	0	$(\alpha_2-\alpha_0)^{-1}(\alpha_2-\alpha_1)^{-1}$	$(\alpha_2-\alpha_1)^{-1}\left[-(\alpha_3-\alpha_0)^{-1}(\alpha_2-\alpha_0)^{-1}-\right.$ $\left.(\alpha_3-\alpha_0)^{-1}(\alpha_3-\alpha_1)^{-1}\right]$ $-(\alpha_3-\alpha_0)^{-1}(\alpha_3-\alpha_1)^{-1}(\alpha_3-\alpha_2)^{-1}$	$-(\alpha_2-\alpha_1)^{-1}\left\{(\alpha_4-\alpha_0)^{-1}(\alpha_3-\alpha_0)^{-1}(\alpha_2-\alpha_0)^{-1}-\right.$ $-(\alpha_3-\alpha_1)^{-1}\left[-(\alpha_4-\alpha_0)^{-1}(\alpha_3-\alpha_0)^{-1}-(\alpha_4-\alpha_0)^{-1}\right.$ $\left.\left.(\alpha_4-\alpha_1)^{-1}\right]\right\}-(\alpha_3-\alpha_2)^{-1}\left\{(\alpha_3-\alpha_1)^{-1}\left[-(\alpha_4-\alpha_0)^{-1}-\right.\right.$ $(\alpha_4-\alpha_0)^{-1}$ $\left.\left.(\alpha_4-\alpha_1)^{-1}\right]-(\alpha_4-\alpha_0)^{-1}(\alpha_4-\alpha_1)^{-1}(\alpha_4-\alpha_2)^{-1}\right\}$
0	0	0	$(\alpha_3-\alpha_0)^{-1}(\alpha_3-\alpha_1)^{-1}(\alpha_3-\alpha_2)^{-1}$	$(\alpha_3-\alpha_2)^{-1}\left\{(\alpha_3-\alpha_1)^{-1}\left[-(\alpha_4-\alpha_0)^{-1}(\alpha_3-\alpha_0)^{-1}-\right.\right.$ $\left.(\alpha_4-\alpha_0)^{-1}(\alpha_4-\alpha_1)^{-1}\right]-(\alpha_4-\alpha_0)^{-1}(\alpha_4-\alpha_1)^{-1}$ $\left.(\alpha_4-\alpha_2)^{-1}\right\}-(\alpha_4-\alpha_0)^{-1}(\alpha_4-\alpha_1)^{-1}(\alpha_4-\alpha_2)^{-1}$ $(\alpha_4-\alpha_3)^{-1}$
0	0	0	0	$(\alpha_4-\alpha_0)^{-1}(\alpha_4-\alpha_1)^{-1}(\alpha_4-\alpha_2)^{-1}(\alpha_4-\alpha_3)^{-1}$

Si $x_n = U_n d^{(n)} = \begin{bmatrix} x_{n-1} \\ 0 \end{bmatrix} + d_n^{(n)} u^{(n)}$ donde la $u^{(n)}$ es la última columna en U_n. Los x_n van cambiando y queda entonces encontrar la manera de ir generando los $u^{(n)}$.

Tomando la última componente de $c^{(n)} = U_n^T f_n$ que es $f[\alpha_0, \alpha_1, \ldots, \alpha_n] = \sum_{k=0}^{n} u_k^{(n)} f_k$ notamos que las $u_k^{(n)}$ expresan las diferencias divididas de orden n-ésimo, es decir

$$u_k^{(n)} = \left[(\alpha_k - \alpha_0)\cdots(\alpha_k - \alpha_{k-1})(\alpha_k - \alpha_{k+1})\cdots(\alpha_k - \alpha_n)\right]^{-1}$$

Utilizando las expresiones anteriores concretamos el siguiente algoritmo.

Algoritmo Progresivo para el sistema Primal

$$x^{(0)} = d^{(0)} = b_0 \; ; \; u_0^{(0)} = 1$$

Para $n = 1,2,3,\cdots$

$$d_n^{(0)} = b_n$$

$$d_n^{(k+1)} = d_n^{(k)} - \alpha_k d_{n-1}^{(k)}; \qquad k = 0,1,\ldots,n-1$$

$$u_n^{(n)} = \left[(\alpha_n - \alpha_0)(\alpha_n - \alpha_1)\cdots(\alpha_n - \alpha_{n-1})\right]^{-1}$$

$$x_n^{(n)} = d_n^{(n)} u_n^{(n)}$$

$$u_k^{(n)} = u_k^{(n-1)}(\alpha_k - \alpha_n)^{-1}; \qquad k = n-1,\ldots,1,0$$

$$x_k^{(n)} = x_k^{(n-1)} + d_n^{(n)} u_k^{(n)}; \qquad k = n-1,\ldots,1,0$$

1.4 Desarrollo de Procedimientos progresivos

Recordando: Parte No. 1 Obtención de las diferencias divididas

$c^{(0)} = f$

Para $k = 0,1,\ldots,n-1$

Para $j = n, n-1, \ldots, k+1$

$$c_j^{(k+1)} = \left(c_j^{(k)} - c_{j-1}^{(k)}\right) \Big/ \left(\alpha_j - \alpha_{j-k-1}\right)$$

Desarrollando:

$k = 0$;

$j = n$; hasta $k+1$

$$c_n^{(1)} = \left(c_n^{(0)} - c_{n-1}^{(0)}\right) \Big/ \left(\alpha_n - \alpha_{n-1}\right)$$

$$c_{n-1}^{(1)} = \left(c_{n-1}^{(0)} - c_{n-2}^{(0)}\right) \Big/ \left(\alpha_{n-1} - \alpha_{n-2}\right)$$

$$c_{n-2}^{(1)} = \left(c_{n-2}^{(0)} - c_{n-3}^{(0)}\right) \Big/ \left(\alpha_{n-2} - \alpha_{n-3}\right)$$

$$\vdots \qquad\qquad \vdots$$

$$c_2^{(1)} = \left(c_2^{(0)} - c_1^{(0)}\right) \Big/ \left(\alpha_2 - \alpha_1\right)$$

$$c_1^{(1)} = \left(c_1^{(0)} - c_0^{(0)}\right) \Big/ \left(\alpha_1 - \alpha_0\right)$$

$k = 1$;

$j = n$; hasta $k+1$

$$c_n^{(2)} = \left(c_n^{(1)} - c_{n-1}^{(1)}\right) \Big/ \left(\alpha_n - \alpha_{n-2}\right)$$

$$c_{n-1}^{(2)} = \left(c_{n-1}^{(1)} - c_{n-2}^{(1)}\right) \Big/ \left(\alpha_{n-1} - \alpha_{n-3}\right)$$

$$c_{n-2}^{(2)} = \left(c_{n-2}^{(1)} - c_{n-3}^{(1)}\right) \Big/ \left(\alpha_{n-2} - \alpha_{n-4}\right)$$

$$\vdots \qquad\qquad \vdots$$

$$c_{2}^{(2)} = \left(c_{2}^{(1)} - c_{1}^{(1)}\right) \Big/ \left(\alpha_{2} - \alpha_{0}\right)$$

$k = 2$;

$j = n$; hasta $k+1$

$$c_{n}^{(3)} = \left(c_{n}^{(2)} - c_{n-1}^{(2)}\right) \Big/ \left(\alpha_{n} - \alpha_{n-3}\right)$$

$$c_{n-1}^{(3)} = \left(c_{n-1}^{(2)} - c_{n-2}^{(2)}\right) \Big/ \left(\alpha_{n-1} - \alpha_{n-4}\right)$$

$$\vdots \qquad\qquad \vdots$$

$$c_{n-2}^{(3)} = \left(c_{n-2}^{(2)} - c_{n-3}^{(2)}\right) \Big/ \left(\alpha_{n-2} - \alpha_{0}\right)$$

$k = 3$;

$j = n$; hasta $k+1$

$$c_{n}^{(4)} = \left(c_{n}^{(3)} - c_{n-1}^{(3)}\right) \Big/ \left(\alpha_{n} - \alpha_{n-4}\right)$$

$$c_{n-1}^{(4)} = \left(c_{n-1}^{(3)} - c_{n-2}^{(3)}\right) \Big/ \left(\alpha_{n-1} - \alpha_{0}\right)$$

$k = 4$;

$j = n$; hasta $k+1$

$$c_{n}^{(5)} = \left(c_{n}^{(4)} - c_{n-1}^{(4)}\right) \Big/ \left(\alpha_{n} - \alpha_{0}\right)$$

$k = n-1$;

$j = n$; hasta $k+1$

$$c_n^{(n)} = \left(c_n^{(n-1)} - c_{n-1}^{(n-1)}\right) \Big/ (\alpha_n - \alpha_0)$$

Para observar con más claridad lo que se obtuvo, vea la tabla de valores calculados en el apartado anterior 1.3.1, pág.37.

Ahora en la nueva versión progresiva haciendo $j = n$

$$\mathrm{c}^{(0)} = \mathrm{f}$$
$$Para\, k = 0, 1, \ldots, n-1$$

$$c_n^{(k+1)} = \left(c_n^{(k)} - c_{n-1}^{(k)}\right) \Big/ (\alpha_n - \alpha_{n-k-1})$$

Al agregar el nuevo valor α_n se observa :

$k = 0$; hasta $n-1$

$$c_n^{(1)} = \left(c_n^{(0)} - c_{n-1}^{(0)}\right) \Big/ (\alpha_n - \alpha_{n-1})$$

$k = 1$; $\quad c_n^{(2)} = \left(c_n^{(1)} - c_{n-1}^{(1)}\right) \Big/ (\alpha_n - \alpha_{n-2})$

$k = 2$; $\quad c_n^{(3)} = \left(c_n^{(2)} - c_{n-1}^{(2)}\right) \Big/ (\alpha_n - \alpha_{n-3})$

$k = 3$; $\quad c_n^{(4)} = \left(c_n^{(3)} - c_{n-1}^{(3)}\right) \Big/ (\alpha_n - \alpha_{n-4})$

$\vdots \qquad\qquad \vdots$

$k = n-1$; $\quad c_n^{(n)} = \left(c_n^{(n-1)} - c_{n-1}^{(n-1)}\right) \Big/ (\alpha_n - \alpha_0)$

Esto es en términos de la interpolación polinomial; en términos de algebra matricial dijimos que los $\mathrm{c}^{(k)}$ son recursivamente generados por

$$\mathrm{c}^{(0)} = \mathrm{f}, \qquad \mathrm{c}^{(k+1)} = D_k^{-1} M_k \mathrm{c}^{(k)}, \qquad M_k = L_k(1), \qquad k = 0, 1, \ldots, n-1$$

Entonces cuando $k = n-1$; llegamos a que $c^{(n)} = D_{n-1}^{-1} M_{n-1} c^{(n-1)}$

Si los primeros $c^{(n-1)}$ permanecen sin cambio, calculamos entonces $c^{(n)}$, para cuando se agrega el nuevo valor α_n.

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix} = \begin{pmatrix} 1 & & & & 0 \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ 0 & & & & (\alpha_n - \alpha_0)^{-1} \end{pmatrix} \begin{pmatrix} 1 & & & & 0 \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ 0 & & & -1 & 1 \end{pmatrix} c^{(n-1)}$$

Recordando $c^{(k)} = \begin{pmatrix} c_0 \\ \vdots \\ c_k \\ f[\alpha_1, \ldots, \alpha_{k+1}] \\ \vdots \\ f[\alpha_{n-k}, \ldots, \alpha_n] \end{pmatrix}$ para $k = 0, 1, \ldots, n,$

entonces cuando $k = n-1$ queda así : $c^{(n-1)} = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ f[\alpha_1, \alpha_2, \ldots, \alpha_n] \end{pmatrix}$

el producto total será

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix} = \begin{pmatrix} 1 & & & & 0 \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ 0 & & & & (\alpha_n - \alpha_0)^{-1} \end{pmatrix} \begin{pmatrix} 1 & & & & 0 \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ 0 & & & -1 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ f[\alpha_1, \alpha_2, \ldots, \alpha_n] \end{pmatrix}$$

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix} = \begin{pmatrix} 1 & & & & 0 \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ 0 & & & & (\alpha_n - \alpha_0)^{-1} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ f[\alpha_1, \alpha_2, \ldots, \alpha_n] - c_{n-1} \end{pmatrix}$$

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ \dfrac{f[\alpha_1, \alpha_2, \ldots, \alpha_n] - c_{n-1}}{\alpha_n - \alpha_0} \end{pmatrix} \quad \text{por lo tanto} \quad c_n = \frac{f[\alpha_1, \alpha_2, \ldots, \alpha_n] - c_{n-1}}{\alpha_n - \alpha_0}$$

Ahora se emplea la Parte No.2 para obtener los coeficientes a_i.

$$\mathrm{a}_n = L_n^T c^{(n)} = \begin{pmatrix} \mathrm{a}_{n-1} \\ 0 \end{pmatrix} + c_n^{(n)} \left(\mathrm{m}^{(n)} \right)^T$$

donde $m^{(n)}$ es el último renglón en la matriz L_n. observamos que

$$L_n^T = \left[\begin{array}{c|c} L_{n-1}^T & \begin{matrix} m_0^{(n)} \\ \vdots \\ m_{n-1}^{(n)} \end{matrix} \\ \hline 0 & m_n^{(n)} \end{array}\right]$$ se compone de $L^T = N_0^T N_1^T \cdots N_{n-1}^T$, donde $N_k = L_k(\alpha_k)$

para $k = n-1, \ldots, 1, 0$. Veamos ahora como se desarrolla

$$N_0^T = \begin{bmatrix} 1 & -\alpha_0 & & & 0 \\ & 1 & -\alpha_0 & & \\ & & 1 & \ddots & \\ & & & \ddots & -\alpha_0 \\ 0 & & & & 1 \end{bmatrix} \quad N_1^T = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ & 1 & -\alpha_1 & & \vdots \\ & & 1 & \ddots & 0 \\ & & & \ddots & -\alpha_1 \\ 0 & & & & 1 \end{bmatrix}$$

$$N_2^T = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ & 1 & 0 & & \vdots \\ & & 1 & -\alpha_2 & 0 \\ & & & 1 & -\alpha_2 \\ 0 & & & & 1 \end{bmatrix} \ldots \; N_{n-2}^T = \begin{bmatrix} 1 & 0 & & \cdots & 0 \\ & \ddots & & & \vdots \\ & & 1 & -\alpha_{n-2} & 0 \\ & & & 1 & -\alpha_{n-2} \\ 0 & & & & 1 \end{bmatrix}$$

$$N_{n-1}^T = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ & 1 & & & \vdots \\ & & \ddots & & 0 \\ & & & 1 & -\alpha_{n-1} \\ 0 & & & & 1 \end{bmatrix}$$

Desarrollando el producto $N_0^T N_1^T \cdots N_{n-1}^T$ queda :

$$L_n^T = \begin{bmatrix} 1 & -\alpha_0 & \alpha_0\alpha_1 & -\alpha_0\alpha_1\alpha_2 & \alpha_0\alpha_1\alpha_2\alpha_3 & -\alpha_0\cdots\alpha_{n-1} \\ & 1 & -\alpha_0-\alpha_1 & \alpha_0\alpha_1+\alpha_1\alpha_2+\alpha_0\alpha_2 & -\alpha_0\alpha_1\alpha_2-\alpha_1\alpha_2\alpha_3-\alpha_0\alpha_2\alpha_3-\alpha_0\alpha_1\alpha_3 & \vdots \\ & & 1 & -\alpha_0-\alpha_1-\alpha_2 & & \vdots \\ & & & 1 & -\alpha_0-\alpha_1-\alpha_2-\alpha_3 & \vdots \\ & & & & \ddots & -\alpha_0\cdots-\alpha_{n-1} \\ & & & & & 1 \\ 0 & & & & & \end{bmatrix}$$

La regla de lo observado es : diagonal principal = unos,
la primera diagonal por encima de la principal es

$$-\alpha_0$$
$$-\alpha_0-\alpha_1$$
$$-\alpha_0-\alpha_1-\alpha_2$$
$$\vdots$$
$$-\alpha_0-\alpha_1-\alpha_2\cdots-\alpha_{n-1}$$

segunda diagonal por encima de la principal es

$$\alpha_0\alpha_1$$
$$\alpha_0\alpha_1+\alpha_1\alpha_2+\alpha_0\alpha_2$$
$$\alpha_0\alpha_1+\alpha_1\alpha_2+\alpha_0\alpha_2+\alpha_0\alpha_3+\alpha_1\alpha_3+\alpha_2\alpha_3$$
$$\alpha_0\alpha_1+\alpha_1\alpha_2+\alpha_0\alpha_2+\alpha_0\alpha_3+\alpha_1\alpha_3+\alpha_2\alpha_3+\ldots+\alpha_0\alpha_{n-1}+\alpha_1\alpha_{n-1}+\alpha_2\alpha_{n-1}+\alpha_3\alpha_{n-1}+\ldots \alpha_{n-1}\alpha_n$$

$$\vdots$$

tercera diagonal por encima de la principal es

$$-\alpha_0\alpha_1\alpha_2$$
$$-\alpha_0\alpha_1\alpha_2-\alpha_1\alpha_2\alpha_3-\alpha_0\alpha_2\alpha_3-\alpha_0\alpha_1\alpha_3$$
$$-\alpha_0\alpha_1\alpha_2-\alpha_1\alpha_2\alpha_3-\alpha_0\alpha_2\alpha_3-\alpha_0\alpha_1\alpha_3-\ldots-\alpha_2\alpha_3\alpha_{n-1}-\alpha_1\alpha_3\alpha_{n-1}-\alpha_1\alpha_2\alpha_{n-1}- -\alpha_0\alpha_3\alpha_{n-1}-\alpha_0\alpha_2\alpha_{n-1}-\alpha_0\alpha_1\alpha_{n-1}$$

$$\vdots$$

cuarta diagonal por encima de la principal es

$$\alpha_0\alpha_1\alpha_2\alpha_3$$
$$\alpha_0\alpha_1\alpha_2\alpha_3+\ldots+\alpha_0\alpha_2\alpha_3\alpha_{n-1}+\alpha_0\alpha_1\alpha_3\alpha_{n-1}+\alpha_0\alpha_1\alpha_2\alpha_{n-1}+\alpha_1\alpha_2\alpha_3\alpha_{n-1}$$

$$\vdots$$

quinta diagonal por encima de la principal es

$$-\alpha_0\alpha_1\alpha_2\alpha_3\alpha_4$$
$$\vdots$$ etc.

Además si observamos la matriz diagonal L_n^T en su última columna, la cual es $m^{(n)}$, donde por ejemplo para una $n=3$,

$$m_0^{(3)} = -\alpha_0\alpha_1\alpha_2$$
$$m_1^{(3)} = \alpha_0\alpha_1 + \alpha_1\alpha_2 + \alpha_0\alpha_2$$
$$m_2^{(3)} = -\alpha_0 - \alpha_1 - \alpha_2$$
$$m_3^{(3)} = 1, \text{ etc.}$$

que no son otra cosa que las entradas L_n^T. También son los coeficientes del polinomio.

$$P_n(z) = m_0^{(n)} + m_1^{(n)}z + \cdots + m_n^{(n)}z^n$$

Y hasta aquí damos por concluido el desarrollo del procedimiento progresivo para el sitema Dual.

A continuación veamos el desarrollo del procedimiento para el algoritmo progresivo del sistema Primal.

La Parte No. 1 ($d^{(n)} = L_n b_n$)de éste algoritmo Primal se observa así

$$d^{(0)} = b$$

For $k = 0, 1, \cdots, n-1$

$$d_j^{(k+1)} = d_j^{(k)} - \alpha_k d_{j-1}^{(k)} \; ; \; j = n, n-1, \cdots, k+1$$
$$= d_j^{(k)} \qquad ; \; j = k, \cdots, 1, 0$$

Repasando esta Parte No. 1. Bajo una pequeña ejecución del programa...

$k=0$ $d_n^{(1)} = d_n^{(0)} - \alpha_0 d_{n-1}^{(0)}$

$d_{n-1}^{(1)} = d_{n-1}^{(0)} - \alpha_0 d_{n-2}^{(0)}$

$d_{n-2}^{(1)} = d_{n-2}^{(0)} - \alpha_0 d_{n-3}^{(0)}$

$d_{n-3}^{(1)} = d_{n-3}^{(0)} - \alpha_0 d_{n-4}^{(0)}$

$k=1$ $d_n^{(2)} = d_n^{(1)} - \alpha_1 d_{n-1}^{(1)}$

$d_{n-1}^{(2)} = d_{n-1}^{(1)} - \alpha_1 d_{n-2}^{(1)}$

$d_{n-2}^{(2)} = d_{n-2}^{(1)} - \alpha_1 d_{n-3}^{(1)}$

$k=2$ $d_n^{(3)} = d_n^{(2)} - \alpha_2 d_{n-1}^{(2)}$

$d_{n-1}^{(3)} = d_{n-1}^{(2)} - \alpha_2 d_{n-2}^{(2)}$

$\vdots \qquad \vdots \qquad \vdots$

$k = n-1$ $d_n^{(n)} = d_n^{(n-1)} - \alpha_{n-1} d_{n-1}^{(n-1)}$

Los primeros $n-1$ componentes permanecen sin cambio alguno, los $d_{n-1}^{(k)}$; $k=0,1,\cdots,n-1$, (en este caso los elementos que se guardan son $d_{n-1}^{(0)},d_{n-1}^{(1)},d_{n-1}^{(2)},\ldots d_{n-1}^{(n-1)}$) y ahora lo único que se tiene que calcular son los $d_n^{(k)}$; $k=0,1,\cdots,n$, los que pueden obtenerse haciendo $j=n$ en dicha Parte No. 1 como sigue

$$
\begin{aligned}
d_n^{(1)} &= d_n^{(0)} - \alpha_0 d_{n-1}^{(0)} \\
d_n^{(2)} &= d_n^{(1)} - \alpha_1 d_{n-1}^{(1)} \\
d_n^{(3)} &= d_n^{(2)} - \alpha_2 d_{n-1}^{(2)} \\
&\vdots \quad\quad \vdots \quad\quad \vdots \\
d_n^{(n-2)} &= d_n^{(n-3)} - \alpha_{n-3} d_{n-1}^{(n-3)} \\
d_n^{(n-1)} &= d_n^{(n-2)} - \alpha_{n-2} d_{n-1}^{(n-2)} \\
d_n^{(n)} &= d_n^{(n-1)} - \alpha_{n-1} d_{n-1}^{(n-1)}
\end{aligned}
$$

Para la parte No. 2 tenemos que $x_n = U_n d^{(n)}$, donde la $U_n = \left[\begin{array}{c|c} U_{n-1} & \begin{matrix} u_0^{(n)} \\ \cdots \\ u_{n-1}^{(n)} \end{matrix} \\ \hline 0 & u_n^{(n)} \end{array}\right]$

$$U_{n-1} = M_0^T D_0^{-1} \cdots M_{n-1}^T D_{n-1}^{-1};\ M_k = L_k(1)\ ;\ D_k = \text{diag}\left(\underbrace{1,\ldots,1}_{k+1},(\alpha_{k+1}-\alpha_0),\ldots,(\alpha_n-\alpha_{n-k-1})\right)$$

planteando las matrices requeridas...

$$M_0 = L_0(1) = \begin{bmatrix} 1 & & & & & 0 \\ -1 & 1 & & & & \\ & -1 & 1 & & & \\ & & -1 & 1 & & \\ & & & \ddots & \ddots & \\ 0 & & & & -1 & 1 \end{bmatrix} \qquad M_0^T = \begin{bmatrix} 1 & -1 & & & & 0 \\ & 1 & -1 & & & \\ & & 1 & -1 & & \\ & & & 1 & & \\ & & & \ddots & \ddots & -1 \\ 0 & & & & & 1 \end{bmatrix}$$

En seguida se ilustran las matrices bidiagonales transpuestas:

$$M_1^T = \begin{bmatrix} 1 & & & & 0 \\ & 1 & -1 & & \\ & & 1 & -1 & \\ & & & 1 & \\ & & & \ddots & \ddots & -1 \\ 0 & & & & & 1 \end{bmatrix} \quad M_2^T = \begin{bmatrix} 1 & & & & 0 \\ & 1 & & & \\ & & 1 & -1 & \\ & & & 1 & -1 \\ & & & \ddots & \ddots & -1 \\ 0 & & & & & 1 \end{bmatrix}$$

$$M_3^T = \begin{bmatrix} 1 & & & & 0 \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & -1 \\ & & & \ddots & \ddots & -1 \\ 0 & & & & & 1 \end{bmatrix} \cdots M_{n-1}^T = \begin{bmatrix} 1 & & & & 0 \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 & -1 \\ 0 & & & & & 1 \end{bmatrix}$$

Ahora las matrices diagonales

$$D_0^{-1} = \begin{bmatrix} 1 & & & & & 0 \\ & (\alpha_1 - \alpha_0)^{-1} & & & & \\ & & (\alpha_2 - \alpha_1)^{-1} & & & \\ & & & (\alpha_3 - \alpha_2)^{-1} & & \\ & & & & \ddots & \\ 0 & & & & & (\alpha_n - \alpha_{n-1})^{-1} \end{bmatrix}$$

$$D_{n-1}^{-1} = \begin{bmatrix} 1 & & & & & 0 \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & \ddots & \\ 0 & & & & & (\alpha_n - \alpha_0)^{-1} \end{bmatrix}.$$

Desarrollamos el producto de matrices para obtener $U_{n-1} = M_0^T D_0^{-1} \cdots M_{n-1}^T D_{n-1}^{-1}$
Recordemos que U_{n-1} es una matriz triangular superior.

CAPITULO 2

CASO CONFLUENTE

2.1.Algoritmos
2.1.1.Dual

Este caso *Confluente* no es tan simplificado como el *No-Confluente* tratado en el primer capítulo del presente trabajo. Aquí el llamado *orden de confluencia*, está determinado por el número de argumentos repetidos. El caso *Confluente* se denota de la siguiente manera

$$V = V(\beta_0,\gamma_0;\beta_1,\gamma_1;\ldots\beta_p,\gamma_p)$$

donde en un sentido general el *orden de confluencia* en los $p+1$ puntos β_i , $i=0,\ldots p$, es (γ_i-1); $i=0,\ldots p$. Para el *No-Confluente* se omite puesto que no tiene repeticiones en los argumentos :$V(\alpha_0,1;\alpha_1,1;\ldots\alpha_n,1)\equiv V(\alpha_0,\alpha_1,\ldots\alpha_n)$

Al igual que en el capítulo primero encontramos la solución tanto para el sistema *Dual* ($V^T\mathrm{a}=\mathrm{f}$) como para el *Primal* ($V\mathrm{x}=\mathrm{b}$).

Consideramos una matriz de Vandermonde

$$V = V(\alpha_0,\alpha_1,\ldots,\alpha_n) = \begin{bmatrix} 1 & 1 & & 1 \\ \alpha_0 & \alpha_1 & \cdots & \alpha_n \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_0^n & \alpha_1^n & \cdots & \alpha_n^n \end{bmatrix}$$

Sustituimos la segunda columna por la diferencia de la segunda y la primera dividida por ε

$$V(\alpha_0,\alpha_0+\varepsilon,\alpha_2,\ldots,\alpha_n)$$

En el límite cuando $\varepsilon\to 0$ obtenemos precisamente el caso *Confluente*

$$V = V(\alpha_0,2;\alpha_2\ldots,\alpha_n) = \begin{bmatrix} 1 & 0 & 1 & \cdots & 1 \\ \alpha_0 & 1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_0^2 & 2\alpha_0 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \alpha_0^n & n\alpha_0^{n-1} & \alpha_2^n & \cdots & \alpha_n^n \end{bmatrix}$$

De igual forma para la matriz de Vandermonde $V(\alpha_0,\ldots,\alpha_{p-1},\alpha_p,\ldots,\alpha_n)$ donde $\alpha_j = \alpha_0 + j\varepsilon$ para $j = 0,1,\ldots,p-1$, podemos reemplazar la columna $(j+1)$ por ε^{-j} veces la j-ésima diferencia de las primeras $(j+1)$ columnas.
Entonces en el límite cuando $\varepsilon \to 0$, la matriz de Vandermonde queda

$$V = V(\alpha_0, p; \alpha_p \ldots, \alpha_n) = \begin{bmatrix} 1 & 0 & 1 & \cdots & 1 \\ \alpha_0 & 1 & \alpha_p & \cdots & \alpha_n \\ \alpha_0^2 & 2\alpha_0 & \alpha_p^2 & \cdots & \alpha_n^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \alpha_0^n & n\alpha_0^{n-1} & \alpha_p^n & \cdots & \alpha_n^n \end{bmatrix}$$

decimos que el orden de la confluencia en α_0 es $p-1$.
Ahora sea $(p+1)$ el número de α_j que son distintas entre sí, $p \le n$; llamamos a estos números $\beta_0,\beta_1,\ldots,\beta_p$. Además denotamos por γ_i el número de argumentos α_j que son iguales a β_i. Asumimos que los α_j se han ordenado de la siguiente manera

$$\alpha_{m_i+k} = \beta_i \text{ para } k = 0,1,\ldots,\gamma_j - 1 \text{ , e } i = 0,1,\ldots,p \qquad (1)$$

en donde hacemos la $m_0 = 0$ y las siguientes como $m_i = \sum_{j=0}^{i-1} \gamma_j$ para $i = 1,\ldots,p+1$

$i = 0$

$k = 0$ $\quad \alpha_{m_0+0} = \beta_0$; $\quad m_0 = 0$

$k = 1$ $\quad \alpha_{m_0+1} = \beta_0$

$\vdots$ $\qquad\qquad \vdots$

$k = \gamma_0 - 1$ $\quad \alpha_{m_0+\gamma_0-1} = \beta_0$

$i = 1$

$k = 0$ $\quad \alpha_{m_1+0} = \beta_1$; $\quad m_1 = \gamma_0$

$k = 1$ $\quad \alpha_{m_1+1} = \beta_1$

$\vdots$ $\qquad\qquad \vdots$

$k = \gamma_1 - 1$ $\quad \alpha_{m_1+\gamma_1-1} = \beta_1$

$i = 2$

$$
\begin{array}{lll}
 & k=0 & \alpha_{m_2+0}=\beta_2\ ; \quad m_2=\gamma_0+\gamma_1 \\
 & k=1 & \alpha_{m_2+1}=\beta_2 \\
 & \vdots & \vdots \\
 & k=\gamma_2-1 & \alpha_{m_2+\gamma_2-1}=\beta_2 \\
\vdots & \vdots & \\
\vdots & \vdots & \\
i=p-1 & & \\
 & k=0 & \alpha_{m_{p-1}+0}=\beta_{p-1}\ ; m_{p-1}=\gamma_0+\gamma_1+\cdots+\gamma_{p-2} \\
 & k=1 & \alpha_{m_{p-1}+1}=\beta_{p-1} \\
 & \vdots & \vdots \\
 & k=\gamma_{p-1}-1 & \alpha_{m_{p-1}+\gamma_{p-1}-1}=\beta_{p-1} \\
i=p & & \\
 & k=0 & \alpha_{m_p+0}=\beta_p\ ; m_p=\gamma_0+\gamma_1+\cdots+\gamma_{p-2}+\gamma_{p-1} \\
 & k=1 & \alpha_{m_p+1}=\beta_p \\
 & \vdots & \vdots \\
 & k=\gamma_p-1 & \alpha_{m_p+\gamma_p-1}=\beta_p
\end{array}
$$

La matriz $V=V\left(\beta_0,\gamma_0;\beta_1,\gamma_1;\ldots\beta_p,\gamma_p\right)$ tiene m_{p+1} columnas y las γ_i columnas correspondientes a las β_i estan dadas por

$$V^{(k)}(\beta_i)=\left(\frac{d^kV(\beta)}{d\beta^k}\right)_{\beta=\beta_i},\ k=0,1,\ldots,\gamma_i-1$$

El siguiente paso es averiguar cómo modificar nuestros algoritmos para resolver $V\,\mathrm{x}=\mathrm{b}$ y $V^T\mathrm{a}=\mathrm{f}$, ahora para el caso confluente.

Empezamos con el sistema Dual ($V^T\mathrm{a}=\mathrm{f}$) donde ahora V es una matriz $V_{p+1}=V\left(\beta_0,\gamma_0;\beta_1,\gamma_1;\beta_2,\gamma_2;\ldots;\beta_p,\gamma_p\right)$.

Consideramos ahora las $n+1$ funciones y sus valores derivados $f_{m_i+k}=f^{(k)}(\beta_i)$ para $k=0,1,\ldots,\gamma_j-1$, e $i=0,1,\ldots,p$ de una función $f(z)$ dada.

Sea $P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ donde $n = m_{p+1} - 1$, el polinomio que resuelve ahora el problema de interpolación, es decir que en el punto β_i , $i = 0,1,\ldots,p$, satisface las condiciones

$$P^{(k)}(\beta_i) = f_{m_i+k} \quad k = 0,1,\ldots,\gamma_i - 1$$

El problema de interpolación se traduce en $P^{(k)}(\beta_j) = f_j^{(k)}$ para $k = 0,1,\ldots,\gamma_j - 1$; $j = 0,1,\ldots,p$.

Entonces los coeficientes del polinomio $P(x)$ satisfacen la transpuesta ($V^T \mathrm{a} = \mathrm{f}$) en el sistema Dual, donde la variable f es denotada por $\mathrm{f} = \left(f_0,\ldots,f_0^{(\gamma_0-1)},f_1,\ldots,f_1^{(\gamma_1-1)},\ldots,f_m,\ldots,f_m^{(\gamma_m-1)}\right)^T$. Este es el método de Newton de interpolación para el caso confluente.

Si nos permitimos $\alpha_{m_j+k} = \beta_j \quad k = 0,1,\ldots,\gamma_j - 1$; $j = 0,1,\ldots,p$. entonces los polinomios fundamentales $P_k(x)$ satisfacen

$$\begin{aligned}
P_0(x) &= 1 \\
P_1(x) &= (x - \alpha_0) \\
P_2(x) &= (x - \alpha_0)(x - \alpha_1) \\
P_3(x) &= (x - \alpha_0)(x - \alpha_1)(x - \alpha_2) \\
&\vdots \\
P_n(x) &= (x - \alpha_0)(x - \alpha_1)(x - \alpha_2)\ldots(x - \alpha_{n-1})
\end{aligned}$$

y no cambian, el cambio conciste en que las *diferencias divididas* van a ser generalizadas.

Dichos polinomios $P_k(x)$ son generados recursivamente por

$$P_0(x) = 1,\ P_k(x) = P_{k-1}(x)(x - \alpha_{k-1}),\ k = 1,\ldots,n$$

Tenemos luego

$$P(x) = c_0 P_0(x) + c_1 P_1(x) + \cdots + c_n P_n(x)$$

que equivale a decir $P(x) = (P_0(x), P_1(x), \cdots, P_n(x))\mathrm{c}_k$

donde $\mathrm{c}_k = f[\alpha_0, \alpha_1, \ldots, \alpha_k]$, $k = 0,1,\ldots,n$, son las *diferencias divididas* obtenidas mediante la recursividad.

Denotaremos las *diferencias divididas* para el caso *Confluente* de la siguiente forma:

$$f[\beta_i,\gamma_0;\beta_{i+1},\gamma_1;\cdots\beta_n,\gamma_n]$$

donde el orden de las *Diferencias Divididas* para el caso confluente es $q-1$, siendo $q=\sum_{i=0}^{n}\gamma_i$ y donde α_i $(\alpha_{m_j+k}=\beta_j,\ k=0,1,\ldots,\gamma_j-1,\ j=0,1,\ldots,p)$ ocurre γ_j veces. Se podría decir que el caso *Confluente* está determinado por el número de repeticiones α_i iguales al argumento β_j. Si un γ_j es igual a uno, se omite. Quedando :

$$f[\beta_i,1;\beta_{i+1},1;\cdots;\beta_n,1]\equiv f[\beta_i,\beta_{i+1},\cdots\beta_n]$$

Hay varios métodos para evaluar estas diferencias divididas.

Si $f^{(\gamma_j-1)}$ existe, entonces : $f[\beta_i,\gamma_j]=\dfrac{f^{(\gamma_j-1)}(\beta_i)}{(\gamma_j-1)!}$,

su fundamento lo refiere a un teorema, para el caso particular, llamado de *Rolle*.

TEOREMA. Supóngase que $f\in[a,b]$ y que $x_0,\ldots,x_n$ son números distintos en $[a,b]$. Entonces existe un número $\xi\in[a,b]$ tal que $f[x_0,x_1,\ldots,x_n]=\dfrac{f^{(n)}(\xi)}{n!}$

DEMOSTRACION. Sea $g(x)=f(x)-P_n(x)$, como $f(x_i)=P_n(x_i)$ para cada $i=0,1,\ldots,n$, g tiene $n+1$ ceros distintos en $[a,b]$. El teorema de Rolle generalizado implica que existe un número ξ en $[a,b]$ con $g^{(n)}(\xi)=0$ así que

$$0=f^{(n)}(\xi)-P_n^{(n)}(\xi)$$

como $P_n(x)$ es un polinomio de grado n cuyo coeficiente principal es $f[x_0,x_1,\ldots,x_n]$ entonces $P_n^{(n)}(x)=f[x_0,x_1,\ldots,x_n]\,n!$ □

Otra manera de evaluar las *Diferencias Divididas* para el caso confluente es obteniendo las del caso *No-Confluente* y haciendo uso de la siguiente relación :

$f[\beta_i,\gamma_0;\beta_{i+1},\gamma_1;\cdots\beta_n,\gamma_n]=$

$$=\frac{1}{(\gamma_0-1)!}\frac{1}{(\gamma_1-1)!}\cdots\frac{1}{(\gamma_n-1)!}\frac{\partial^{\gamma_0+\gamma_1+\cdots+\gamma_n-n-1}}{\partial\beta_i^{\gamma_0-1}\partial\beta_{i+1}^{\gamma_1-1}\cdots\partial\beta_n^{\gamma_n-1}}f[\beta_0,\beta_1,\ldots,\beta_n]$$

Las diferencias divididas para el caso confluente de orden k-ésimo pueden computarse recursivamente $f\left[\alpha_{j-k-1},\ldots,\alpha_j\right]=\dfrac{f\left[\alpha_{j-k},\ldots,\alpha_j\right]-f\left[\alpha_{j-k-1},\ldots,\alpha_{j-1}\right]}{\alpha_j-\alpha_{j-k-1}}$, ésta relación es válida cuando $\alpha_j \neq \alpha_{j-k-1}$. Por otro lado, del ordenamiento $\alpha_{m_j+k}=\beta_j$, llegamos a que $\alpha_j=\alpha_{j-k-1}$ y en este caso se deduce que los argumentos α_i son iguales, entonces su diferencia dividida será

$$f\left[\alpha_j,k+1\right]=f\underbrace{\left[\alpha_{j-k-1},\cdots,\alpha_j\right]}_{k+1\text{ veces}}=f^{(k)}\left(\alpha_j\right)/k!$$

y el cálculo de tal diferencia no será tomado en cuenta. Recordemos que es un cociente el cálculo de la diferencia dividida y no esta definida la división entre 0. Definimos los vectores $c^{(k)}$ para los valores $k=0,1,\ldots,n$ con anterioridad en el caso No-confluente como el vector de las diferencias divididas $c^{(k)}=\left(c_0,\ldots,c_k,f\left[\alpha_1,\ldots,\alpha_{k+1}\right],\ldots,f\left[\alpha_{n-k},\ldots,\alpha_n\right]\right)^T$. Entonces el vector de las diferencias divididas $c=c^{(n)}$ se computa en el algoritmo Dual, paso No.1.

A continuación se ilustra el algoritmo para obtener las *diferencias divididas* $(c=c^{(n)})$ en el caso *Dual Confluente*

Dual Paso no.1
$c=f$
for $k=0$ step 1 until $n-1$ do
begin $t=\max\left(\gamma_p-k,1\right)$
for $i=p$ step -1 until 0 do
if $m_{i+1}>k+1$
begin $s=\max\left(m_i,k+1\right)$; $r=\max\left(\gamma_{i-1}-k,1\right)$
for $j=m_{i+1}-1$ step -1 until s do
if $j>m_{i+1}-t$ $c_j=c_j/(k+1)$ else
if $j\neq s$ $\left(c_j-c_{j-1}\right)/\left(\alpha_j-\alpha_{j-k-1}\right)$ else
$\left(c_j-c_{j-r}\right)/\left(\alpha_j-\alpha_{j-k-1}\right)$
$t=r$
end
end

Tratamos ahora de describir dicho algoritmo...

Vamos a observar que todos los componentes $f_{m_j+k} = f^{(k)}(\beta_j)$; $j = 0,1,\ldots,p$; $k = 0,1,\ldots,\gamma_j - 1$, que son las funciones dadas y sus valores derivados de f están divididos en $p+1$ grupos, donde el i -ésimo grupo tiene γ_i-miembros α_i correspondiente al mismo argumento β_i. Se comenta que la división entre $\alpha_j - \alpha_{j-k-1}$ algunas veces no se efectuará, por lo del acomodo de $\alpha_j = \beta_i$, pero $\alpha_{j-k-1} \neq \beta_i$ cuando se observe que

$$j \leq m_{i+1} - \max(\gamma_i - k, 1) \leq m_i + k$$

La iteración más interna inicia siempre con el valor de $j = m_{i+1} - 1$ y luego lo va decrementando hasta un valor mínimo que es la variable $s = \max(m_i, k+1)$. La iteración intermedia, contiene al indice "i " el cual toma valores hasta $p+1$, para idicar el número del grupo en proceso. Y la iteración más externa o última en incrementarse será la de la variable k la cual toma valores desde 0 hasta $n-1$. Dicha iteración llevará el conteo de las k-tandas de diferencias divididas. Se puede observar que en el algoritmo una variable clave son los γ_i- renglones (por que hablamos del caso $V^T \mathrm{a} = \mathrm{f}$). Se definió inicialmente $m_0 = 0$ y las siguientes como $m_i = \sum_{j=0}^{i-1} \gamma_j$ para $i = 1,\ldots,p+1$, vamos, la variable m_i contiene las sumas de las γ_i, por grupos y total final, y esta variable servirá para indicar el índice del inicio y término de los límites de cada grupo en un contexto matricial.

Regresando al contexto de la primera iteración o la más externa, cuando $k = 0$ equivale a decir que vamos a obtener la primera tanda $c_j^{(k)}$ de diferencias divididas, o sea $c_j = {f_{m_{i+1}-1} - f_{m_{i+1}-1-1}}/{\alpha_j - \alpha_{j-k-1}}$; analizando el numerador de éste cociente, j en la iteración más interna toma el valor $j = m_{i+1} - 1$ donde m_{i+1} es igual al número de polinomios dados $P^{(k)}(\beta_j)$; $j = 0,1,\ldots,p$; $k = 0,1,\ldots,\gamma_j - 1$, para el caso en la que la iteración intermedia inicia con $i = p$.

En resumen, tomará al inicio del procedimiento mecanizado las funciones $f^0(x)$ de cada uno de los diferentes $p+1$ grupos en orden decreciente, es decir hará la diferencia en el numerador con los valores f de los polinomios (sin derivar) que contengan igual grado.

Y en lo que respecta al denominador, los subíndices j y $j-k-1$, en esta tanda se ejecuta la diferencia del denominador de la misma manera que el caso no-confluente $(\alpha_n-\alpha_{n-1},\alpha_{n-1}-\alpha_{n-2},\alpha_{n-2}-\alpha_{n-3},\ldots,\alpha_1-\alpha_0)$ solo que dicha división se hará cuando sean diferentes los α's.

La variable j decrece partiendo desde el valor n hasta el $k+1$, cuando $k=0$ se calculan las diferencias $c_n,c_{n-1},\ldots,c_1$; con $k=1$ se calculan $c_n,c_{n-1},\ldots,c_2$; con $k=2$ se calculan $c_n,c_{n-1},\ldots,c_3$; hasta el tope final que es cuando toma el último valor $k=n-1$ se calcula la última diferencia c_n.

Haciendo referencia al caso No-confluente este se formularía a partir del valor de $k=q-1$ donde q toma el valor máximo de entre todas las γ_i's. Es por eso que hay que observar una regla importante: tomar el orden máximo de confluencia (recordemos que está determinado por el número de argumentos repetidos desde el planteamiento inicial del problema), esto es:

$$q=\max\left\{\gamma_j-1\right\}$$

entonces se podrá decir en cuantas k-tandas están afectadas las diferencias divididas. A partir de la tanda $k+1$, las diferencias divididas se obtienen de igual forma que el caso No-confluente.

Definimos los vectores $c^{(k)}$ para los valores $k=0,1,\ldots,n$ por

$$c^{(k)}=\left(c_0,\ldots,c_k,f[\alpha_1,\ldots,\alpha_{k+1}],\ldots f[\alpha_{n-k},\ldots,\alpha_n]\right)^T$$

Una vez que se han obtenido las c utilizamos recursión mediante el esquema de Horner a fin de que se pueda determinar las a. Para ello introducimos los vectores $a^{(k)}=\left(c_0,\ldots,c_{k-1},a_k^{(k)},\ldots,a_n^{(k)}\right)$, entonces se computan $a^{(n)}=c$ y $a^{(0)}=a$ mediante el paso No. 2 del algoritmo Dual

Dual paso no. 2

$a = c$

for $k = n-1$ step -1 until 0 do

for $j = k$ step 1 until $n-1$ do

$$a_j = a_j - a_k \times a_{j+1}$$

Por lo tanto, en conclusión final una vez obtenidas las diferencias divididas y habiendo utilizado el esquema de Homer llegamos a la obtención de los valores a_i del polinomio descrito en el planteamiento inicial $P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ donde $n = m_{p+1} - 1$.

2.1.Algoritmos
2.1.2.Primal

Para derivar un algoritmo que resuelva el sistema $V\mathrm{x}=\mathrm{b}$, utilizamos las fórmulas que se implementaron para el sistema dual, nada mas que ahora en lenguaje matricial como a continuación se muestra:

$$\mathrm{c}^{(0)}=\mathrm{f},\ \mathrm{c}^{(k+1)}=D_k^{-1}M_k\mathrm{c}^{(k)} \text{ para } k=0,1,\ldots,n-1 \qquad (3)$$
$$\mathrm{a}^{(n)}=\mathrm{c},\ \mathrm{a}^{(k)}=N_k^T\mathrm{a}^{(k+1)} \text{ para } k=n-1,\ldots,1,0$$

D_k es la matriz diagonal, M_k es la matriz triangular inferior unitaria y N_k^T es la triangular superior y bidiagonal. Estas matrices se utilizaron en el caso No-confluente. Para el caso Confluente las matrices N_k permancecen inalteradas y las D_k y M_k sufren modificaciones como luego se verá.

De (3) el método de Newton para interpolación para resolver $V^T\mathrm{a}=\mathrm{f}$ puede ser planteado de la siguiente forma

$$\mathrm{c}=U^T\mathrm{f},\qquad \mathrm{a}=L^T\mathrm{c}$$

donde con $U^T=D_{n-1}^{-1}M_{n-1}\ldots D_0^{-1}M_0$ y $L^T=N_0^TN_1^T\cdots N_{n-1}^T$ obtenemos $\mathrm{a}=L^TU^T\mathrm{f}$, y de aquí que $V^{-T}=L^TU^T$, consecuentemente $\mathrm{a}=\left(V^{-1}\right)\mathrm{f}$ donde $V^{-1}=UL$. Y entonces quitamos la transposición, quedando ahora la $U=M_0^TD_0^{-1}\ldots M_{n-1}^TD_{n-1}^{-1}$ y la $L=N_{n-1}\cdots N_0$, y si $q=\max\left\{\gamma_j-1\right\}$ es el orden máximo de confluencia, según lo descrito en el apartado anterior, la presentación de U en el caso de confluencia :

$$U=(M_0^{'})^T\left(D_0^{'}\right)^{-1}\cdots(M_{q-1}^{'})^T\left(D_{q-1}^{'}\right)^{-1}(M_q)^T\left(D_q^{-1}\right)\cdots(M_{n-1})^T\left(D_{n-1}^{-1}\right)$$

donde hay la matriz ahora llamada $M_k^{'}$ es igual a la M_k excepto en los $q-k+1$ renglones conprendidos en $i+k,\cdots,i+q$; y en las matrices diagonales $D_k^{'}$, hay cambio en los renglones $i+k,\cdots,i+q-1$ con respecto a las D_k.

La relación de recurrencia para darle solución a $Vx = b$ es :

$$x = x^{(0)}, \qquad g^{(0)} = b, \qquad g^{(k+1)} = N_k g^{(k)} \text{ para } k = 0,1,\ldots,n-1$$
$$x^{(n)} = g^{(n)}, \qquad x^{(k)} = M_k^T D_k^{-1} x^{(k+1)} \text{ para } k = n-1,\ldots,1,0$$

En base a estas relaciones presentamos los dos pasos de los que se compone dicho proceso.

Primal Paso no.1

$g = b$

for $k = 0$ step 1 until $n-1$ do

for $j = n$ step -1 until $k+1$ do

$g_j = g_j - \alpha_k \times g_{j-1}$

Este paso es exactamente el mismo que en el caso No-confluente.

Primal Paso no.2

$x = g$

for $k = n-1$ step -1 until 0 do

for $i = 0$ step 1 until p do

if $m_{i+1} > k+1$

begin $s = \max(m_i, k+1)$

$t = \max(\gamma_i - k, 1)$

if $m_i \le k+1$ $r = 1$

for $j = s$ step 1 until m_{i+1} -1 do

if $j > m_{i+1} - t$

$x_j = x_j / (k+1)$ else

$x_j = x_j / (\alpha_j - \alpha_{j-k-1})$

for $j = s$ step 1 until m_{i+1} - t do

if $j \neq s$

$x_{j-1} = x_{j-1} - x_j$ else

$x_{j-r} = x_{j-r} - x_j$

$r = t$

end

2.2.*Desarrollo de Procedimientos*

Cuando resolvemos el sistema Dual ó $V^T \mathrm{a} = \mathrm{f}$ sabemos que $\mathrm{a} = V^{-T}\mathrm{f}$ y además que una matriz inversa se puede descomponer en el producto de dos matrices triangulares, una inferior y una superior, $V^{-T} = L^T U^T$. Aquí la triangular inferior $L^T = N_0^T N_1^T \cdots N_{n-1}^T$ permanece sin cambio con respecto el caso No-confluente, y la triangular superior ahora tiene una variante; en el No-confluente se describe dicha matriz como $U^T = D_{n-1}^{-1} M_{n-1} \cdots D_0^{-1} M_0$ y ahora incluye nuevas matrices $M_k^{'}$ y $D_k^{'}$ quedando así:

$$U^T = \left(D_{n-1}^{-1}\right) M_{n-1} \cdots \left(D_q^{-1}\right) M_q \left(D_{q-1}^{'}\right)^{-1} M_{q-1}^{'} \cdots \left(D_0^{'}\right)^{-1} M_0^{'}$$

Las matrices $M_k^{'}$ y M_k son iguales a excepción de los renglones $i+k \ldots i+q$ estos, $q-k+1$ renglones se conforman de la siguiente manera

$$\begin{bmatrix} 0 & \cdots & 0 & 1 & \cdots & \cdots & \cdots & 0 \\ & & & 0 & 1 & & & \\ & & & & \ddots & & & \\ 0 & \cdots & -1 & \cdots & 0 & 1 & \cdots & 0 \end{bmatrix}$$

También se definió $q = \max\left\{\gamma_j - 1\right\}$ como el orden máximo de confluencia, la variable $k = 0, 1, \ldots \gamma_j - 1$; y la variable $i = 0, \ldots p$, donde p es el número de bloques de α_j que son distintas entre sí.

La primera parte del sistema Primal $V\,\mathrm{x} = \mathrm{b}$, en términos matriciales es :

$$x = x^{(0)}$$
$$g^{(0)} = b, \quad g^{(k+1)} = N_k g^{(k)} \text{ para } k = 0, 1, \ldots, n-1$$

donde $N_k = L_k(\alpha_k)$. En seguida ilustramos la manipulación a esta primera parte :

$g^{(1)} = N_0 g^{(0)}$

o lo que es lo mismo $g^{(1)} = \begin{bmatrix} 1 & & & & 0 \\ -\alpha_0 & 1 & & & \\ & -\alpha_0 & 1 & & \\ & & -\alpha_0 & 1 & \\ 0 & & & \cdots & \cdots \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \cdots \\ b_n \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 - \alpha_0 b_0 \\ b_2 - \alpha_0 b_1 \\ \cdots \\ b_n - \alpha_0 b_{n-1} \end{bmatrix}$

$$g^{(2)} = \begin{bmatrix} 1 & & & & 0 \\ 0 & 1 & & & \\ & -\alpha_1 & 1 & & \\ & & -\alpha_1 & 1 & \\ 0 & & & \cdots & \cdots \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 - \alpha_0 b_0 \\ b_2 - \alpha_0 b_1 \\ \cdots \\ b_n - \alpha_0 b_{n-1} \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 - \alpha_0 b_0 \\ b_2 - \alpha_0 b_1 - \alpha_1 (b_1 - \alpha_0 b_0) \\ \cdots \\ b_n - \alpha_0 b_{n-1} - \alpha_1 (b_{n-1} - \alpha_0 b_{n-2}) \end{bmatrix}$$

$$g^{(3)} = \begin{bmatrix} 1 & & & & 0 \\ 0 & 1 & & & \\ & 0 & 1 & & \\ & & -\alpha_2 & 1 & \\ 0 & & & \cdots & \cdots \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 - \alpha_0 b_0 \\ b_2 - \alpha_0 b_1 - \alpha_1 (b_1 - \alpha_0 b_0) \\ \cdots \\ b_n - \alpha_0 b_{n-1} - \alpha_1 (b_{n-1} - \alpha_0 b_{n-2}) \end{bmatrix} =$$

$$\begin{bmatrix} b_0 \\ b_1 - \alpha_0 b_0 \\ b_2 - \alpha_0 b_1 - \alpha_1 (b_1 - \alpha_0 b_0) \\ \cdots \\ b_n - \alpha_0 b_{n-1} - \alpha_1 (b_{n-1} - \alpha_0 b_{n-2}) - \alpha_2 \left[b_{n-1} - \alpha_0 b_{n-2} - \alpha_1 (b_{n-2} - \alpha_0 b_{n-3}) \right] \end{bmatrix}$$

Así sucesivamente hasta obtener los $g^{(n)}$:

$$\begin{aligned} g^{(1)} &= N_0 g^{(0)} \\ g^{(2)} &= N_1 g^{(1)} \\ &\vdots \\ g^{(n)} &= N_{n-1} g^{(n-1)} \end{aligned}$$

En otras palabras obtuvimos $g^{(n)} = N_{n-1} N_{n-2} \cdots N_1 N_0 b$.

A continuación repasamos el algoritmo computacional

$g = b$

for $k = 0$ step 1 until $n-1$ do

for $j = n$ step -1 until $k+1$ do

$g_j = g_j - \alpha_k \times g_{j-1}$

Desarrollando

$$
\begin{aligned}
g_n &= g_n - \alpha_0 \times g_{n-1} \\
g_{n-1} &= g_{n-1} - \alpha_0 \times g_{n-2} \\
&\vdots \\
g_3 &= g_3 - \alpha_0 \times g_2 \\
g_2 &= g_2 - \alpha_0 \times g_1 \\
g_1 &= g_1 - \alpha_0 \times g_0 \\
g_n &= g_n - \alpha_1 \times g_{n-1} \\
g_{n-1} &= g_{n-1} - \alpha_1 \times g_{n-2} \\
&\vdots \\
g_3 &= g_3 - \alpha_1 \times g_2 \\
g_2 &= g_2 - \alpha_1 \times g_1 \\
g_n &= g_n - \alpha_2 \times g_{n-1} \\
g_{n-1} &= g_{n-1} - \alpha_2 \times g_{n-2} \\
&\vdots \\
g_3 &= g_3 - \alpha_2 \times g_2 \\
&\vdots \\
g_n &= g_n - \alpha_{n-1} \times g_{n-1}
\end{aligned}
$$

Por lo tanto se llegó a la solución

$$g_n = g_n - \alpha_{n-1}\left[g_{n-1} - \alpha_{n-2}\left[\cdots \alpha_2\left[g_2 - \alpha_1\left[g_1 - \alpha_0 g_0\right]\right]\cdots\right]\right]$$

que nos servirá de datos de entrada para la segunda parte.

La segunda parte equivale a:

$\mathbf{x}^{(n)} = \mathbf{g}^{(n)}$, $\quad \mathbf{x}^{(k)} = M_k^T D_k^{-1} \mathbf{x}^{(k+1)}$ $\quad$ para $k = n-1, \ldots, 1, 0$

evaluando la recursividad :

$$
\begin{aligned}
\mathbf{x}^{(n-1)} &= M_{n-1}^T D_{n-1}^{-1} \mathbf{x}^{(n)} \\
\mathbf{x}^{(n-2)} &= M_{n-2}^T D_{n-2}^{-1} \mathbf{x}^{(n-1)} \\
&\vdots \\
\mathbf{x}^{(1)} &= M_1^T D_1^{-1} \mathbf{x}^{(2)} \\
\mathbf{x}^{(0)} &= M_0^T D_0^{-1} \mathbf{x}^{(1)}
\end{aligned}
$$

Y de esta manera obtenemos los valores del sistema $x = V^{-1}b$ donde $V^{-1} = UL$.

CAPITULO 3

EJEMPLOS

3. Ejemplos

Ejemplo No. 1. Interpolación Polinomial

A continuación presentamos un ejemplo para poder ilustrar el Algoritmo de Newton para Interpolación Polinomial. Consideramos los puntos p_0 (1.3, 24.8), p_1(2.5,13.2), p_2 (1.8,11.1), p_3 (3.9,17.0), y tenemos la tabla

x	01.3	02.5	01.8	03.9
y	24.8	13.2	11.1	17.0

Recordando: $p(x_i) = y_i$

$p_0(x) = 24.8$

$p_1(x) = p_0(x) + c(x - x_0)$

$= 24.8 + c(x - 1.3)$

$13.2 = 24.8 + c(2.5-1.3)$

$13.2 - 24.8 = c(2.5-1.3)$

$-11.6 = c(1.2) \Rightarrow c = -11.6/1.2 \approx -9.666$

$p_2(x) = p_0(x) + c(x - x_0) + c'(x - x_0)(x - x_1)$

$= 24.8 - 9.66(x-1.3) + c'(x-1.3)(x-2.5)$

$11.1 = 24.8 - 9.66(1.8-1.3) + c'(1.8-1.3)(1.8-2.5)$

$11.1 - 24.8 = -9.66(0.5) + c'(0.5)(-0.7)$

$-13.7 = -4.83 + c'(-0.35)$

$-13.7 + 4.83 = -c'(0.35) \Rightarrow c' = -8.87 / -0.35 \Rightarrow c' = 25.34$

$p_3(x) = p_0(x) + c(x - x_0) + c'(x - x_0)(x - x_1) + c''(x - x_0)(x - x_1)(x - x_2)$

$= 24.8 + (-9.666)(x-1.3) + 25.34(x-1.3)(x-2.5) + c''(x-1.3)(x-2.5)(x-1.8)$

$17.0 = 24.8 - 9.666(3.9-1.3) + 25.34(3.9-1.3)(3.9-2.5) + c''(3.9-1.3)(3.9-2.5)(3.9-1.8)$

$17.0 - 24.8 = -9.666(2.6) + 25.34(2.6)(1.4) + c''(2.6)(1.4)(2.1)$

$17.0 - 24.8 = -25.132 + 92.238 + c''(7.644)$

$17.0 - 24.8 + 25.132 - 92.238 = c''(7.644)$

$-74.906 = c''(7.644) \Rightarrow c'' = -74.906 / 7.644 = -9.799$

Ejemplo No. 2. Caso No-Confluente (Dual)

Presentamos el sistema de ecuaciones lineales para resolver :

$$\begin{aligned} a_0 + a_1 + a_2 + a_3 &= 10 \\ a_0 + 2a_1 + 4a_2 + 8a_3 &= 26 \\ a_0 + 3a_1 + 9a_2 + 27a_3 &= 58 \\ a_0 + 4a_1 + 16a_2 + 64a_3 &= 112 \end{aligned} \qquad (*)$$

pare ello utilizamos el sistema $V^T \mathbf{a} = \mathbf{f}$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 64 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 26 \\ 58 \\ 112 \end{bmatrix}$$

aplicando el algoritmo Dual en su primera parte ($\mathbf{c} = U^T \mathbf{f}$; $\mathbf{c}^{(0)} = \mathbf{f}$), tenemos:

$$c_3^{(1)} = \frac{c_3^{(0)} - c_2^{(0)}}{\alpha_3 - \alpha_2} = \frac{112 - 58}{4 - 3} = 54$$

$$c_2^{(1)} = \frac{c_2^{(0)} - c_1^{(0)}}{\alpha_2 - \alpha_1} = \frac{58 - 26}{3 - 2} = 32$$

$$c_1^{(1)} = \frac{c_1^{(0)} - c_0^{(0)}}{\alpha_1 - \alpha_0} = \frac{26 - 10}{2 - 1} = 16$$

$$c_3^{(2)} = \frac{c_3^{(1)} - c_2^{(1)}}{\alpha_3 - \alpha_1} = \frac{54 - 32}{4 - 2} = 11$$

$$c_2^{(2)} = \frac{c_2^{(1)} - c_1^{(1)}}{\alpha_2 - \alpha_0} = \frac{32 - 16}{3 - 1} = 8$$

$$c_3^{(3)} = \frac{c_3^{(2)} - c_2^{(2)}}{\alpha_3 - \alpha_0} = \frac{11 - 8}{4 - 1} = 1$$

y de ésta manera obtuvimos las diferencias divididas $c_1^{(1)} = 16$, $c_2^{(2)} = 8$, $c_3^{(3)} = 1$ y $c_0^{(0)} = 10$ la cual está tomada del planteamiento inicial dado que es el punto inicial de partida.

Ahora pasamos a la segunda parte del algoritmo Dual a fin de encontrar finalmente los coeficientes del polinomio buscados

$$P(x)=(1,(x-\alpha_0),(x-\alpha_0)(x-\alpha_1),(x-\alpha_0)(x-\alpha_1)(x-\alpha_2))(10,16,8,1)$$

según la notación $P(x)=(q_0(x),q_1(x),q_2(x),\cdots,)c$ donde c_k son las diferencias divididas. Aplicamos la segunda parte o esquema de Horner.

$$c_2^{(2)}=c_2^{(2)}-c_3^{(3)}x_2 \qquad =8-1(3)=5$$

$$c_1^{(1)}=c_1^{(1)}-c_2^{(2)}x_1 \qquad =16-5(2)=6$$
$$c_2^{(2)}=c_2^{(2)}-c_3^{(3)}x_1 \qquad =5-1(2)=3$$

$$c_0^{(0)}=c_0^{(0)}-c_1^{(1)}x_0 \qquad =10-6(1)=4$$
$$c_1^{(1)}=c_1^{(1)}-c_2^{(2)}x_0 \qquad =6-3(1)=3$$
$$c_2^{(2)}=c_2^{(2)}-c_3^{(3)}x_0 \qquad =3-1(1)=2$$

$\therefore$ desarrolando $P(x)=10+16(x-1)+8(x-1)(x-2)+1(x-1)(x-2)(x-3)$

$$=4+3x+2x^2+1x^3$$

Por lo tanto la solución del sistema (*) es :

$$a_0=4$$
$$a_1=3$$
$$a_2=2$$
$$a_3=1$$

Ejemplo No. 3. Caso No-Confluente (Primal)

Resolvemos el sistema de ecuaciones lineales

$$\begin{aligned} z_0 + z_1 + z_2 + z_3 &= 0 \\ z_0 + 2z_1 + 3z_2 + 4z_3 &= -1 \\ z_0 + 4z_1 + 9z_2 + 16z_3 &= 3 \\ z_0 + 8z_1 + 27z_2 + 64z_3 &= 35 \end{aligned} \qquad (**)$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 64 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 3 \\ 35 \end{bmatrix}$$

Para ello planteamos el sistema Primal $V\mathrm{z} = \mathrm{b}$.
Aplicando el primer paso del algoritmo Primal tenemos:

$$\begin{aligned} b_4 &= b_4 - x_1 b_3 && = 35 - 1(3) = 32 \\ b_3 &= b_3 - x_1 b_2 && = 3 - 1(-1) = 4 \\ b_2 &= b_2 - x_1 b_1 && = -1 - 1(0) = -1 \\ b_4 &= b_4 - x_2 b_3 && = 32 - 2(4) = 24 \\ b_3 &= b_3 - x_2 b_2 && = 4 - 2(-1) = 6 \\ b_4 &= b_4 - x_3 b_3 && = 24 - 3(6) = 6 \end{aligned}$$

$\therefore b_1 = 0, b_2 = -1, b_3 = 6, b_4 = 6$ éstas son las diferencias divididas. Ahora procedemos a la segunda parte del algoritmo y a obtener los coeficientes del polinomio :

$$\begin{aligned} b_4 &= b_4 / x_4 - x_0 && = 6/(4-1) = 2 \\ b_3 &= b_3 - b_4 && = 6 - 2 = 4 \\ b_3 &= b_3 / x_3 - x_0 && = 4/(3-1) = 2 \\ b_4 &= b_4 / x_4 - x_1 && = 2/(4-2) = 1 \\ b_2 &= b_2 - b_3 && = -1 - 2 = -3 \\ b_3 &= b_3 - b_4 && = 2 - 1 = 1 \\ b_2 &= b_2 / x_2 - x_0 && = -3/1 = -3 \\ b_3 &= b_3 / x_3 - x_1 && = 1/1 = 1 \\ b_4 &= b_4 / x_4 - x_2 && = 2/2 = 1 \\ b_1 &= b_1 - b_2 && = 0 - (-3) = 3 \\ b_2 &= b_2 - b_3 && = -3 - 1 = -4 \\ b_3 &= b_3 - b_4 && = 1 - 1 = 0 \end{aligned}$$

$$
\begin{aligned}
b_1 &= b_1 / x_1 - x_0 && = 3/1 \\
b_2 &= b_2 / x_2 - x_1 && = -4/1 \\
b_3 &= b_3 / x_3 - x_2 && = 0/1 \qquad \therefore b_1 = 3, b_2 = -4 \\
b_4 &= b_4 / x_4 - x_3 && = 1/1 \qquad \text{y } b_3 = 0, b_4 = 1
\end{aligned}
$$

Por lo tanto la solución del sistema (**) es :

$$
\begin{aligned}
z_0 &= 3 \\
z_1 &= -4 \\
z_2 &= 0 \\
z_3 &= 1
\end{aligned}
$$

Ejemplo No. 4. Caso No-Confluente (Dual-Progresivo)

Presentamos el sistema de ecuaciones lineales por resolver :

$$\begin{aligned} a_0 + a_1 + a_2 &= 10 \\ a_0 + 2a_1 + 4a_2 &= 26 \qquad (*) \\ a_0 + 3a_1 + 9a_2 &= 58 \end{aligned}$$

para ello utilizamos el sistema $V^T \mathrm{a} = \mathrm{f}$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 9 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 10 \\ 26 \\ 58 \end{bmatrix}$$

aplicando el algoritmo Dual en su primera parte ($\mathrm{c} = U^T \mathrm{f}$; $\mathrm{c}^{(0)} = \mathrm{f}$), tenemos:

$$c_2^{(1)} = \frac{c_2^{(0)} - c_1^{(0)}}{\alpha_2 - \alpha_1} = \frac{58 - 26}{3 - 2} = 32$$

$$c_1^{(1)} = \frac{c_1^{(0)} - c_0^{(0)}}{\alpha_1 - \alpha_0} = \frac{26 - 10}{2 - 1} = 16$$

$$c_2^{(2)} = \frac{c_2^{(1)} - c_2^{(1)}}{\alpha_2 - \alpha_0} = \frac{32 - 16}{3 - 1} = 8$$

y de ésta manera obtuvimos las diferencias divididas , $c_1^{(1)} = 16$, $c_2^{(2)} = 8$. y $c_0^{(0)} = 10$ que que nos es dada como punto inicial en el enunciado del sistema.

Ahora se agrega un nuevo valor $\alpha_3 = 4$ con su respectivo $c_3^{(0)} = 112$

$$\begin{aligned} a_0 + a_1 + a_2 + a_3 &= 10 \\ a_0 + 2a_1 + 4a_2 + 8a_3 &= 26 \\ a_0 + 3a_1 + 9a_2 + 27a_3 &= 58 \\ a_0 + 4a_1 + 16a_2 + 64a_3 &= 112 \end{aligned}$$

lo ponemos en la forma $V^T \mathrm{a} = \mathrm{f}$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 64 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 26 \\ 58 \\ 112 \end{bmatrix}$$

Una vez agregado el nuevo valor α_n, se conservan las últimas diferencias divididas o sea $c_{n-1}^{(k)}, k = 0,1,...n-1$, en otras palabras se guardan los últimos valores obtenidos, para el caso es el primer renglón del siguiente esquema

$$c_2^{(0)}, c_2^{(1)}, c_2^{(2)}$$
$$c_1^{(0)}, c_1^{(1)}$$
$$c_0^{(0)}$$

y los valores que se obtienen son los $c_3^{(1)}, c_3^{(2)}, c_3^{(3)}$, los valores agregados son $\alpha_3 = 4$ y $c_3^{(0)} = 112$ haciendo $j = n$ en el algoritmo Dual paso No.1 se computa:

$$c_3^{(1)} = \frac{c_3^{(0)} - c_2^{(0)}}{\alpha_3 - \alpha_2} = \frac{112-58}{4-3} = 54$$

$$c_3^{(2)} = \frac{c_3^{(1)} - c_2^{(1)}}{\alpha_3 - \alpha_1} = \frac{54-32}{4-2} = 11$$

$$c_3^{(3)} = \frac{c_3^{(2)} - c_2^{(2)}}{\alpha_3 - \alpha_0} = \frac{11-8}{4-1} = 1$$

ya se habían obtenido se supone, los valores $c_1^{(1)} = 16$ y $c_2^{(2)} = 8$; $c_0^{(0)} = 10$ que fué valor dado, y ahora se obtuvo $c_3^{(3)} = 1$.

De otra forma, si del paso No.1 del Dual se siguió que los vectores $c^{(k)}$ son recursivamente generados por

$$c^{(0)} = f, \qquad c^{(k+1)} = D_k^{-1} M_k c^{(k)} \quad , k = 0,1,\cdots,n-1$$

en donde la $M_k = L_k(1), \quad D_k = \text{diag}\{1,\cdots,1,(\alpha_{k+1} - \alpha_0),\cdots,(\alpha_n - \alpha_{n-k-1})\}$

y además cuando se agrega un nuevo valor α_n, los primeros $n-1$ componentes en $c^{(n)}$ permanecen sin cambio alguno, entonces habrá que calcular únicamente el nuevo valor $c_n^{(n)}$.

A decir verdad, los valores que ya se han almacenado son $c^{(0)} = 10$ que fué un valor dado, y los $c^{(1)} = 16, c^{(2)} = 8$, y calculamos $c^{(3)}$.

$$c^{(3)} = D_2^{-1} M_2 c^{(2)} \quad \text{entonces} \quad c^{(3)} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & (\alpha_3 - \alpha_0)^{-1} \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{vmatrix} \begin{vmatrix} 10 \\ 16 \\ 8 \\ f[\alpha_1,\alpha_2,\alpha_3] \end{vmatrix}$$

$$c^{(3)} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & (\alpha_3 - \alpha_0)^{-1} \end{vmatrix} \begin{vmatrix} 10 \\ 16 \\ 8 \\ f[\alpha_1,\alpha_2,\alpha_3] - 8 \end{vmatrix} \quad \text{luego} \quad c^{(3)} = \begin{vmatrix} 10 \\ 16 \\ 8 \\ \dfrac{f[\alpha_1,\alpha_2,\alpha_3] - 8}{\alpha_3 - \alpha_0} \end{vmatrix}$$

por otro lado $f[\alpha_1,\alpha_2,\alpha_3] = \dfrac{f[\alpha_2,\alpha_3] - f[\alpha_1,\alpha_2]}{\alpha_3 - \alpha_1}$

y $\quad f[\alpha_1,\alpha_2] = \dfrac{f[\alpha_2] - f[\alpha_1]}{\alpha_2 - \alpha_1} = \dfrac{58-26}{3-2} = 32$

$$f[\alpha_2,\alpha_3] = \frac{f[\alpha_3] - f[\alpha_2]}{\alpha_3 - \alpha_2} = \frac{112-58}{4-3} = 54 \quad \therefore f[\alpha_1,\alpha_2,\alpha_3] = \frac{54-32}{4-2} = 11$$ y ya para finalizar sustituimos

$$c^{(3)} = \left|\begin{array}{c} 10 \\ 16 \\ 8 \\ \hline 11-8 \\ \hline 4-1 \end{array}\right| \text{ o lo que es lo mismo } c^{(3)} = \begin{vmatrix} 10 \\ 16 \\ 8 \\ 1 \end{vmatrix}$$

Y ahora iniciamos el paso No. 2 del algoritmo Dual, el cual para el método de los progresivos se traduce en esto:

$$a_n = L_n^T c^{(n)}$$

para el caso $L_3^T = N_0^T N_1^T N_2^T$ en dónde $N_k = L_k(\alpha_k)$ $k = n-1,\ldots,1,0$, entonces

$$N_0^T = \begin{bmatrix} 1 & -\alpha_0 & 0 & 0 \\ 0 & 1 & -\alpha_0 & 0 \\ 0 & 0 & 1 & -\alpha_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad N_1^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\alpha_1 & 0 \\ 0 & 0 & 1 & -\alpha_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad N_2^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -\alpha_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

obteniendo

$$L_3^T = \left[\begin{array}{ccc|c} 1 & -\alpha_0 & \alpha_0\alpha_1 & -\alpha_0\alpha_1\alpha_2 \\ 0 & 1 & -\alpha_0-\alpha_1 & \alpha_1\alpha_2+\alpha_0\alpha_1+\alpha_0\alpha_2 \\ 0 & 0 & 1 & -\alpha_0-\alpha_1-\alpha_2 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \text{ evaluándola } L_3^T = \left[\begin{array}{ccc|c} 1 & -1 & 2 & -6 \\ 0 & 1 & -3 & 11 \\ 0 & 0 & 1 & -6 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

entonces $\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 2 & -6 \\ 0 & 1 & -3 & 11 \\ 0 & 0 & 1 & -6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 16 \\ 8 \\ 1 \end{bmatrix}$ efectuando el producto $\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}$

Ejemplo No. 5. Caso No-Confluente (Primal-Progresivo)

Resolver el siguiente sistema :

$$z_0 + z_1 + z_2 = 0$$
$$z_0 + 2z_1 + 3z_2 = -1$$
$$z_0 + 4z_1 + 9z_2 = 3$$

que en términos matriciales es $Vz = b$:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 9 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 3 \end{bmatrix}$$

Luego se calculan los valores

$$d_2^{(1)} = d_2^{(0)} - \alpha_0 d_1^{(0)} \quad = 3 - 1(-1) = 4$$
$$d_1^{(1)} = d_1^{(0)} - \alpha_0 d_0^{(0)} \quad = -1 - 1(0) = -1$$
$$d_2^{(2)} = d_2^{(1)} - \alpha_1 d_1^{(1)} \quad = 4 - 2(-1) = 6$$

Ahora se agrega un nuevo valor $\alpha_3 = 4$ con su respectivo $d_3^{(0)} = 35$,

$$z_0 + z_1 + z_2 + z_3 = 0$$
$$z_0 + 2z_1 + 3z_2 + 4z_3 = -1$$
$$z_0 + 4z_1 + 9z_2 + 16z_3 = 3 \qquad (*)$$
$$z_0 + 8z_1 + 27z_2 + 64z_3 = 35$$

que visto matricialmente es :

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 64 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 3 \\ 35 \end{bmatrix}$$

Cuando se agrega el nuevo valor se conservan los valores anteriores $d_{n-1}^{(k)}, k = 0,1,\ldots,n-1$, para el caso en particular $d_2^{(0)}, d_2^{(1)}, d_2^{(2)}$ se almacenan, y $d_3^{(1)}, d_3^{(2)}, d_3^{(3)}$ se calculan. Aplicando el algoritmo computacional:

$$d_3^{(0)} = 35$$
$$d_3^{(1)} = d_3^{(0)} - \alpha_0 d_2^{(0)} \quad = 35 - 1(3) = 32$$

$$d_3^{(2)} = d_3^{(1)} - \alpha_1 d_2^{(1)} = d_3^{(1)} - \alpha_1 (d_2^{(0)} - \alpha_0 d_1^{(0)}) \quad = 32 - 2[3 - 1(-1)] = 24$$

$$d_3^{(3)} = d_3^{(2)} - \alpha_2 d_2^{(2)} \quad = 24 - 3(6) = 6$$

a título de comprobación obtenemos la diferencia dividida anterior

$$d_2^{(2)} = d_2^{(1)} - \alpha_1 d_1^{(1)} = d_2^{(0)} - \alpha_0 d_1^{(0)} - \alpha_1 (d_1^{(0)} - \alpha_0 d_0^{(0)})$$

A continuación el desarrollo del algortimo computacional.

Los datos de entrada $\alpha_i = 1,2,3,4$; $b_i = 0,-1,3,35$ para $i = 0,1,2,3$

$$x^{(0)} = d^{(0)} = b_0 \qquad u_0^{(0)} = 1$$

para $n = 1$

$$d_1^{(0)} = b_1 = -1$$

$$d_1^{(1)} = d_1^{(0)} - \alpha_0 d_0^{(0)} \quad = -1 - 1(0) = -1$$

$$u_1^{(1)} = \frac{1}{(\alpha_1 - \alpha_0)} \quad = \frac{1}{(2-1)} = 1$$

$$x_1^{(1)} = d_1^{(1)} u_1^{(1)} \quad = -1 \cdot 1 = -1$$

$$u_0^{(1)} = u_0^{(0)} (\alpha_0 - \alpha_1)^{-1} \quad = \frac{1}{(1-2)} = -1$$

$$x_0^{(1)} = x_0^{(0)} + d_1^{(1)} u_0^{(1)} \quad = 0 + (-1)(-1) = 1$$

para $n = 2$ $\quad d_2^{(0)} = b_2 = 3$

$$d_2^{(1)} = d_2^{(0)} - \alpha_0 d_1^{(0)} \quad = 3 - 1(-1) = 4$$

$$d_2^{(2)} = d_2^{(1)} - \alpha_1 d_1^{(1)} \quad = 4 - 2(-1) = 6$$

$$u_2^{(2)} = \frac{1}{(\alpha_2 - \alpha_0)(\alpha_2 - \alpha_1)} \quad = \frac{1}{(3-1)(3-2)} = \frac{1}{2}$$

$$x_2^{(2)} = d_2^{(2)} u_2^{(2)} \quad = 6 \cdot 1/2 = 3$$

$$u_1^{(2)} = u_1^{(1)} (\alpha_1 - \alpha_2)^{-1} \quad = \frac{1}{(2-3)} = -1$$

$$x_1^{(2)} = x_1^{(1)} + d_2^{(2)} u_1^{(2)} \quad = -1 + 6(-1) = -7$$

$$u_0^{(2)} = u_0^{(1)} (\alpha_0 - \alpha_2)^{-1} \quad = \frac{-1}{(1-3)} = \frac{1}{2}$$

$$x_0^{(2)} = x_0^{(1)} + d_2^{(2)} u_0^{(2)} \quad = 1 + 6(1/2) = 4$$

para $n = 3$ $\quad d_3^{(0)} = b_3 = 35$

$$d_3^{(1)} = d_3^{(0)} - \alpha_0 d_2^{(0)} \quad = 35 - 1(3) = 32$$

$$d_3^{(2)} = d_3^{(1)} - \alpha_1 d_2^{(1)} \qquad = 32 - 2(4) = 24$$

$$d_3^{(3)} = d_3^{(2)} - \alpha_2 d_2^{(2)} \qquad = 24 - 3(6) = 6$$

$$u_3^{(3)} = \frac{1}{(\alpha_3 - \alpha_0)(\alpha_3 - \alpha_1)(\alpha_3 - \alpha_2)} = \frac{1}{(4-1)(4-2)(4-3)} = \frac{1}{6}$$

$$x_3^{(3)} = d_3^{(3)} u_3^{(3)} \quad = 6 \cdot 1/6 = 1$$

$$u_2^{(3)} = u_2^{(2)}(\alpha_2 - \alpha_3)^{-1} \quad = 1/2 \frac{1}{(3-4)} = -\frac{1}{2}$$

$$x_2^{(3)} = x_2^{(2)} + d_3^{(3)} u_2^{(3)} \qquad = 3 + 6(-1/2) = 0$$

$$u_1^{(3)} = u_1^{(2)}(\alpha_1 - \alpha_3)^{-1} \quad = -1 \frac{1}{(2-4)} = \frac{1}{2}$$

$$x_1^{(3)} = x_1^{(2)} + d_3^{(3)} u_1^{(3)} \qquad = -7 + 6(1/2) = -4$$

$$u_0^{(3)} = u_0^{(2)}(\alpha_0 - \alpha_3)^{-1} \quad = \frac{1}{2} \frac{1}{(1-4)} = -\frac{1}{6}$$

$$x_0^{(3)} = x_0^{(2)} + d_3^{(3)} u_0^{(3)} \qquad = 4 + 6(-1/6) = 3$$

$\therefore \quad x_0^{(3)} = 3, \quad x_1^{(3)} = -4, \quad x_2^{(3)} = 0, \quad x_3^{(3)} = 1$

Por lo tanto estos valores son la solución al sistema (*) en las respectivas variables z_0, z_1, z_2, z_3 .

Ejemplo No. 6. Caso Confluente (Dual)

A continuación presentamos los datos para desarrollar un ejemplo confluente.

$$\alpha_0 = \alpha_1 = \alpha_2 = \beta_0 \qquad \alpha_3 = \alpha_4 = \beta_1 \qquad \alpha_5 = \beta_2$$
$$n = 5, \quad p = 2, \quad \gamma_0 = 3, \quad \gamma_1 = 2, \quad \gamma_2 = 1$$

Otra manera de representar el planteamiento del ejemplo es $V_5 = V[\beta_0, 3; \beta_1, 2; \beta_2]$

$$V(\beta_0,\beta_0,\beta_0,\beta_1,\beta_1,\beta_2) = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ \beta_0 & 1 & 0 & \beta_1 & 1 & \beta_2 \\ \beta_0^2 & 2\beta_0 & 2 & \beta_1^2 & 2\beta_1 & \beta_2^2 \\ \beta_0^3 & 3\beta_0^2 & 6\beta_0 & \beta_1^3 & 3\beta_1^2 & \beta_2^3 \\ \beta_0^4 & 4\beta_0^3 & 12\beta_0^2 & \beta_1^4 & 4\beta_1^3 & \beta_2^4 \\ \beta_0^5 & 5\beta_0^4 & 20\beta_0^3 & \beta_1^5 & 5\beta_1^4 & \beta_2^5 \end{bmatrix}$$

Haciendo referencia al sistema *Dual*, $V^T \mathrm{a} = \mathrm{f}$, sabemos que $\mathrm{f} = \left(f(\beta_0), f'(\beta_0), f''(\beta_0), f(\beta_1), f'(\beta_1), f(\beta_2)\right)$, lo cual quiere decir que el sistema de ecuaciones lineales es:

$$1a_0 + a_1\beta_0 + a_2\beta_0^2 + a_3\beta_0^3 + a_4\beta_0^4 + a_5\beta_0^5 = f(\beta_0)$$
$$1a_1 + 2a_2\beta_0 + 3a_3\beta_0^2 + 4a_4\beta_0^3 + 5a_5\beta_0^4 = f'(\beta_0)$$
$$2a_2 + 6a_3\beta_0 + 12a_4\beta_0^2 + 20a_5\beta_0^3 = f''(\beta_0)$$
$$1a_0 + a_1\beta_1 + a_2\beta_1^2 + a_3\beta_1^3 + a_4\beta_1^4 + a_5\beta_1^5 = f(\beta_1)$$
$$1a_1 + 2a_2\beta_1 + 3a_3\beta_1^2 + 4a_4\beta_1^3 + 5a_5\beta_1^4 = f'(\beta_1)$$
$$1a_0 + a_1\beta_2 + a_2\beta_2^2 + a_3\beta_2^3 + a_4\beta_2^4 + a_5\beta_2^5 = f(\beta_2)$$

Asignando valores a los argumentos $\beta_0 = 1$, $\beta_1 = 2$ y $\beta_2 = 3$ el sistema queda :

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 2 & 6 & 12 & 20 \\ 1 & 2 & 4 & 8 & 16 & 32 \\ 0 & 1 & 4 & 12 & 32 & 80 \\ 1 & 3 & 9 & 27 & 81 & 243 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} f(\beta_0) \\ f'(\beta_0) \\ f''(\beta_0) \\ f(\beta_1) \\ f'(\beta_1) \\ f(\beta_2) \end{bmatrix}$$; tomamos los valores $\mathrm{f} = (-1,4,5,6,7,8)^T$

Los dos primeros valores que toma la variable $k = 0,1$ estan afectados por la confluencia, si $q = \max\{\gamma_j - 1\}$ es el orden máximo de confluencia según lo descrito en el capítulo anterior, $q = 2$. Y ahora a fin de encontrar las diferencias divididas resolvemos $\mathrm{c} = U^T \mathrm{f}$; donde ahora la variante en la cadena U^T es de la siguiente manera : $U^T = (D_{n-1}^{-1}) M_{n-1} \cdots (D_q^{-1}) M_q (D_{q-1}')^{-1} M_{q-1}' \cdots (D_0')^{-1} M_0'$, donde la matriz ahora llamada M_k' es igual a la M_k excepto en algunos renglones. Veamos el caso particular de la M_0' y la M_1'

$$M_0' = \left[\begin{array}{c|c|c} 1 & & \\ & 1 & \\ & & 1 \\ \hline -1 & 1 & \\ & & 1 \\ \hline & -1 & 1 \end{array}\right] \qquad M_1' = \left[\begin{array}{c|c|c} 1 & & \\ & 1 & \\ & & 1 \\ \hline -1 & 1 & \\ & -1 & 1 \\ \hline & & -1 \quad 1 \end{array}\right]$$

Vemos que las matrices están particionadas por bloques correspondientes al número de elementos que tiene cada argumento $\beta_0, \beta_1, \beta_2$ respectivamente.
Y las matrices bidiagonales involucradas en estos pasos son:

$$D_0' = diag(1,1,1,(\alpha_3 - \alpha_2),1,(\alpha_5 - \alpha_4))$$

$$D_1' = diag\left(1,1,\frac{1}{2},(\alpha_3 - \alpha_1),(\alpha_4 - \alpha_2),(\alpha_5 - \alpha_3)\right)$$

Inicialmente se procede a obtener las diferencias divididas. En expresión algebraica esto equivale a escribir $c = U^T f$. Para este caso en particular será : $c = D_4^{-1} M_4 \cdots (D_1')^{-1} M_1' (D_0')^{-1} M_0' f$

$$M_0' f = \left[\begin{array}{c|c|c} \begin{matrix} 1 & & \\ & 1 & \\ & & 1 \end{matrix} & & \\ \hline \begin{matrix} -1 & \\ & \end{matrix} & \begin{matrix} 1 & \\ & 1 \end{matrix} & \\ \hline & \begin{matrix} -1 & 1 \end{matrix} & \end{array}\right] \left[\begin{matrix} f(\beta_0) \\ f'(\beta_0) \\ f''(\beta_0) \\ f(\beta_1) \\ f'(\beta_1) \\ f(\beta_2) \end{matrix}\right] = \left[\begin{matrix} f(\beta_0) \\ f'(\beta_0) \\ f''(\beta_0) \\ f(\beta_1) - f(\beta_0) \\ f'(\beta_1) \\ f(\beta_2) - f(\beta_1) \end{matrix}\right]$$

el siguiente paso en la cadena $c = U^T f$ es efectuar el producto por $(D_0')^{-1}$ y resulta

$$(D_0')^{-1} M_0' f = \left[\begin{matrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & (\alpha_3 - \alpha_2)^{-1} & & \\ & & & & 1 & \\ & & & & & (\alpha_5 - \alpha_4)^{-1} \end{matrix}\right] \left[\begin{matrix} f(\beta_0) \\ f'(\beta_0) \\ f''(\beta_0) \\ f(\beta_1) - f(\beta_0) \\ f'(\beta_1) \\ f(\beta_2) - f(\beta_1) \end{matrix}\right] =$$

$$\left[\begin{matrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & f_3 - f_0/(\alpha_3 - \alpha_2) & & \\ & & & & 1 & \\ & & & & & f_5 - f_3/(\alpha_5 - \alpha_4) \end{matrix}\right]$$

El siguiente paso es hacer el producto con la siguiente matriz en la cadena U^T,

siendo $M_1' = \left[\begin{array}{ccc|cc} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ \hline & -1 & & 1 & \\ & & & -1 & 1 \\ \hline & & & & -1 \end{array}\right]$ entonces el producto siguiente queda :

$$\left[\begin{array}{ccc|cc} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ \hline & -1 & & 1 & \\ & & & -1 & 1 \\ \hline & & & & -1 \end{array}\right] (D_0')^{-1} M_0' f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 - f_1 \\ f_4 - f_3 \\ f_5 - f_4 \end{bmatrix}$$

Hasta este punto se ha calculado la cadena $M_1'(D_0')^{-1} M_0' f$, ahora hay que efectuar el siguiente producto de la matriz en turno de la cadena U^T, sigue $(D_1')^{-1}$. Ilustramos :

$$\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \frac{1}{2} & & & \\ & & & (\alpha_3 - \alpha_1)^{-1} & & \\ & & & & (\alpha_4 - \alpha_2)^{-1} & \\ & & & & & (\alpha_5 - \alpha_3)^{-1} \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 - f_1 \\ f_4 - f_3 \\ f_5 - f_4 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ 1/2\, f_2 \\ f_3 - f_1/\alpha_3 - \alpha_1 \\ f_4 - f_3/\alpha_4 - \alpha_2 \\ f_5 - f_4/\alpha_5 - \alpha_3 \end{bmatrix}$$

la cadena evaluada hasta aquí viene siendo $(D_1')^{-1}M_1'(D_0')^{-1}M_0'f$ que es la parte que tiene confluencia. A partir de aquí en adelante el procedimiento es como el del caso No-confluente.

Tomando en cuenta únicamente estos dos valores, los datos se transforman como el cuadro que ilustramos a continuación:

j	α_j	f_j	$c_j^{(1)}$	$c_j^{(2)}$
0	β_0	$f(\beta_0)$	$c_0 = f(\beta_0)$	
1	β_0	$f'(\beta_0)$	$c_1 = \frac{c_1 - c_0}{\alpha_1 - \alpha_0} = f'(\beta_0)$	
2	β_0	$f''(\beta_0)$	$c_2 = \frac{c_2 - c_1}{\alpha_2 - \alpha_1} = f''(\beta_0)$	$c_2 = \frac{c_2 - c_1}{\alpha_2 - \alpha_0} = \frac{1}{2}f''(\beta_0)$ (*)
3	β_1	$f(\beta_1)$	$c_3 = \frac{c_3 - c_0}{\alpha_3 - \alpha_2} = \frac{f(\beta_1) - f(\beta_0)}{\beta_1 - \beta_0} = f[\beta_1, \beta_0]$	$c_3 = \frac{c_3 - c_1}{\alpha_3 - \alpha_1} = f[\beta_1, \beta_0, \beta_0]$
4	β_1	$f'(\beta_1)$	$c_4 = \frac{c_4 - c_3}{\alpha_4 - \alpha_3} = f'(\beta_1)$	$c_4 = \frac{c_4 - c_3}{\alpha_4 - \alpha_2} = f[\beta_1, \beta_1, \beta_0]$
5	β_2	$f(\beta_2)$	$c_5 = \frac{c_5 - c_3}{\alpha_5 - \alpha_4} = \frac{f(\beta_2) - f(\beta_1)}{\beta_2 - \beta_1} = f[\beta_2, \beta_1]$	$c_5 = \frac{c_5 - c_4}{\alpha_5 - \alpha_3} = f[\beta_2, \beta_1, \beta_1]$

(*) si $f^{(\gamma_j - 1)}$ existe, entonces $f[x_{i-j}, \gamma_j] = \frac{f^{(\gamma_j - 1)}(x_{i-j})}{(\gamma_j - 1)!}$

Los valores siguientes no son afectados por la confluencia y quedan como :

j	α_j	$c_j^{(3)}$	$c_j^{(4)}$	$c_j^{(5)}$
0	β_0			
1	β_0			
2	β_0			
3	β_1	$c_3 = \frac{c_3 - c_2}{\alpha_3 - \alpha_0}$		
4	β_1	$c_4 = \frac{c_4 - c_3}{\alpha_4 - \alpha_1}$	$c_4 = \frac{c_4 - c_3}{\alpha_4 - \alpha_0}$	
5	β_2	$c_5 = \frac{c_5 - c_4}{\alpha_5 - \alpha_2}$	$c_5 = \frac{c_5 - c_4}{\alpha_5 - \alpha_1}$	$c_5 = \frac{c_5 - c_4}{\alpha_5 - \alpha_0}$

Para continuar con la cadena U^T a fin de encontrar las restantes diferencias divididas $c = D_4^{-1} M_4 \cdots (D_1')^{-1} M_1' (D_0')^{-1} M_0' f$ necesitamos definir las siguientes matrices involucradas en el proceso que son $D_4^{-1}, M_4, D_3^{-1}, M_3, D_2^{-1}, M_2$.

Los siguientes pasos no son afectados por la confluencia entonces las matrices quedan como en el caso no confluente

$$M_2 = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & -1 & 1 & & \\ & & & -1 & 1 & \\ & & & & -1 & 1 \end{bmatrix} \quad M_3 = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & -1 & 1 & \\ & & & & -1 & 1 \end{bmatrix} \quad M_4 = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & -1 & 1 \end{bmatrix}$$

y las siguientes bidiagonales que también son utilizadas

$$D_2 = diag(1,1,1,(\alpha_3 - \alpha_0),(\alpha_4 - \alpha_1),(\alpha_5 - \alpha_2))$$

$$D_3 = diag(1,1,1,1,(\alpha_4 - \alpha_0),(\alpha_5 - \alpha_1))$$

$$D_4 = diag(1,1,1,1,1,(\alpha_5 - \alpha_0))$$

Ejemplo No. 7. Caso Confluente (Primal)

Partiendo del mismo planteamiento del ejemplo No. 6, ahora lo resolvemos para el caso *Primal* $V\mathrm{x}=\mathrm{b}$, o sea $\mathrm{x}=V^{-1}\mathrm{b}$, sabemos que la matriz $V^{-1}=UL$. El sistema es

$$
\begin{aligned}
&1+\beta_0+\beta_0^2+\beta_0^3+\beta_0^4+\beta_0^5=f(\beta_0)\\
&1+2\beta_0+3\beta_0^2+4\beta_0^3+5\beta_0^4=f'(\beta_0)\\
&2+6\beta_0+12\beta_0^2+20\beta_0^3=f''(\beta_0)\\
&1+\beta_1+\beta_1^2+\beta_1^3+\beta_1^4+\beta_1^5=f(\beta_1)\\
&1+2\beta_1+3\beta_1^2+4\beta_1^3+5\beta_1^4=f'(\beta_1)\\
&1+\beta_2+\beta_2^2+\beta_2^3+\beta_2^4+\beta_2^5=f(\beta_2)
\end{aligned}
$$

la matriz $U=(M_0')^T(D_0')^{-1}\cdots(M_{q-1}')^T(D_{q-1}')^{-1}(M_q)^T(D_q^{-1})\cdots(M_{n-1})^T(D_{n-1}^{-1})$, y la matriz triangular inferior $L=N_{n-1}\cdots N_0$.

En el caso del sistema Primal, únicamente los dos últimos pasos son influenciados por la confluencia, a saber

$$
x^{(0)}=(M_0')^T(D_0')^{-1}x^{(1)}, \qquad x^{(1)}=(M_1')^T(D_1')^{-1}x^{(2)}
$$

Las matrices D_k' y M_k' para $k=0,1$ quedan de la siguiente manera :

$$
(M_0')^T=\begin{bmatrix}1&0&0&-1&0&0\\0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&1&0&-1\\0&0&0&0&1&0\\0&0&0&0&0&1\end{bmatrix}
\quad \text{y} \quad
(M_1')^T=\begin{bmatrix}1&0&0&0&0&0\\0&1&0&-1&0&0\\0&0&1&0&0&0\\0&0&0&1&-1&0\\0&0&0&0&1&-1\\0&0&0&0&0&1\end{bmatrix}
$$

las matrices restantes M_2^T, M_3^T, M_4^T y M_5^T tienen igual representación que en el caso No-confluente, salvo que ahora son transpuestas. Estas matrices contienen la bidiagonal elemental, ahora las de la diagonal :

$$(D_0')^{-1} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & (\alpha_3 - \alpha_2)^{-1} & & & \\ & & & & 1 & & \\ & & & & & (\alpha_5 - \alpha_4)^{-1} & \end{bmatrix}$$

$$(D_1')^{-1} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \frac{1}{2} & & & & \\ & & & (\alpha_3 - \alpha_1)^{-1} & & & \\ & & & & (\alpha_4 - \alpha_2)^{-1} & & \\ & & & & & (\alpha_5 - \alpha_3)^{-1} & \end{bmatrix}$$

las matrices restantes D_2^{-1}, D_3^{-1}, D_4^{-1} y D_5^{-1} se representan de igual forma que en el caso No-confluente.

En definitiva, la resolución de este caso en particular será :

$$x = V^{-1}b = (M_0')^T (D_0')^{-1} \cdots (M_1')^T (D_1')^{-1} (M_2)^T (D_2^{-1}) \cdots (M_5)^T (D_5^{-1}) N_5 \cdots N_0 \, b$$

```
C*********************************************************C
C     programa INTERPOL.FOR                               C
C     OBJETIVO : Evaluar un polinomio de grado 9 para que C
C                se aproxime a la funcion f(X)=SIN(X) en  C
C                el intervalo [0,Pi/2), 3.1416/2=1.5708,  C
C                el intervalo [0,Pi/6), 3.1416/6=0.5235 y C
C                un conjunto de 10 puntos cubriendo el    C
C                intervalo.
C                x(i)=(i-1)h para h=0.1875 e 1<=i<=10
C*********************************************************C
      DIMENSION X(10),Y(10)
      DATA T,S/1.57079,0.52359/
      OPEN(6,FILE='INTERPOL.SAL',STATUS='NEW')
      write(6,111)
111   format(3x,' valores de entrada (x,f(x))')
      DO 20 I = 1,10
      X(I) = FLOAT(I-1) * 0.1875
      write(6,1111) i,x(i)
1111  format(3x,' x(',i2,') = ',f10.5)
      Y(I) = SIN(X(I))
      write(6,2222) i,y(i)
2222  format(20x,' y(',i2,') = ',f10.5)
20    continue
C
      CALL COEF(10,X,Y)
C
      P1 = EVAL(10,X,Y,T)
      P2 = EVAL(10,X,Y,S)
      write(6,222)
222   format(3x,'Resultados de Pi/2  y Pi/6 respectivamente')
      WRITE(6,101)P1,P2
      WRITE(*,101)P1,P2
101   FORMAT(3X,//,2F20.15)
      stop
      END
C*********************************************************C
C     SUBRUTINA COEF                                      C
C     OBJETIVO : Generar los coeficientes de InterpolaciÂnC
C                Polinomial de Newton                     C
C     ENTRADA : Valores tabulares de X, Y                 C
C     SALIDA : Coeficientes almacenados en Y              C
C*********************************************************C
      SUBROUTINE COEF(N,X,Y)
      DIMENSION X(N),Y(N)
      write(6,3333)
3333  format(3x,//,'Coeficientes de InterpolaciÂn Polinomial')
      DO 20 J = 1, N-1
      write(6,7777) j
7777  format(3x,/,' j =',i2)
      DO 20 I = 1, N-J
      Y(N-I+1) = (Y(N-I+1) - Y(N-I))/(X(N-I+1) - X(N-I-J+1))
      write(6,9999) n-i+1,n-i+1,n-i,
     1              n-i+1,n-i-j+1
9999  format(3x,' y(',i2,') = (y(',i2,')-y(',i2,'))/(x(',i2,')-x(',
     1                                              i2,'))')
      write(6,4444) n-i+1,y(n-i+1)
4444  format(35x,' y(',i2,') = ',f10.5)
20    continue
      RETURN
      END
```

```
C*************************************************************C
C     FUNCTION EVAL                                           C
C     OBJETIVO : Toma las X como nodos para InterpolaciÀn,C
C              las Y como los coeficientes de interpolacion
C              Polinomial de Newton.                          C
C              Este polinomio es evaluado en T                C
C     ENTRADA : Valores reales de X, Y, y un valor T          C
C     SALIDA : Valor T retornado por la funcion               C
C*************************************************************C
      FUNCTION EVAL(N,X,Y,T)
      DIMENSION X(N),Y(N)
      EVAL = Y(N)
      DO 30 I = 1, N-1
      write(6,6666) eval
6666  format(3x,/,'eval = ',f10.5)
      EVAL = EVAL * (T - X(N-I)) + Y(N-I)
      write(6,5555) eval,t,n-i,x(n-i),n-i,y(n-i)
5555  format(3x,/,'eval = ',f10.5,'*( t = ',f10.5,'-x(',i2,') = ',
     1                    f10.5,' )+y(',i2,') = ',f10.5)
30    continue
      RETURN
      END
```

```
valores de entrada (x,f(x))
x( 1) =     .00000
                y( 1) =     .00000
x( 2) =     .18750
                y( 2) =     .18640
x( 3) =     .37500
                y( 3) =     .36627
x( 4) =     .56250
                y( 4) =     .53330
x( 5) =     .75000
                y( 5) =     .68164
x( 6) =     .93750
                y( 6) =     .80608
x( 7) =    1.12500
                y( 7) =     .90227
x( 8) =    1.31250
                y( 8) =     .96683
x( 9) =    1.50000
                y( 9) =     .99749
x(10) =    1.68750
                y(10) =     .99320


Coeficientes de InterpolaciÂn Polinomial

 j = 1
   y(10) = (y(10)-y( 9))/(x(10)-x( 9))
                                    y(10) =    -.02292
   y( 9) = (y( 9)-y( 8))/(x( 9)-x( 8))
                                    y( 9) =     .16356
   y( 8) = (y( 8)-y( 7))/(x( 8)-x( 7))
                                    y( 8) =     .34431
   y( 7) = (y( 7)-y( 6))/(x( 7)-x( 6))
                                    y( 7) =     .51299
   y( 6) = (y( 6)-y( 5))/(x( 6)-x( 5))
                                    y( 6) =     .66369
   y( 5) = (y( 5)-y( 4))/(x( 5)-x( 4))
                                    y( 5) =     .79113
   y( 4) = (y( 4)-y( 3))/(x( 4)-x( 3))
                                    y( 4) =     .89083
   y( 3) = (y( 3)-y( 2))/(x( 3)-x( 2))
                                    y( 3) =     .95930
   y( 2) = (y( 2)-y( 1))/(x( 2)-x( 1))
                                    y( 2) =     .99415

 j = 2
   y(10) = (y(10)-y( 9))/(x(10)-x( 8))
                                    y(10) =    -.49729
   y( 9) = (y( 9)-y( 8))/(x( 9)-x( 7))
                                    y( 9) =    -.48200
   y( 8) = (y( 8)-y( 7))/(x( 8)-x( 6))
                                    y( 8) =    -.44981
   y( 7) = (y( 7)-y( 6))/(x( 7)-x( 5))
                                    y( 7) =    -.40186
   y( 6) = (y( 6)-y( 5))/(x( 6)-x( 4))
                                    y( 6) =    -.33982
   y( 5) = (y( 5)-y( 4))/(x( 5)-x( 3))
                                    y( 5) =    -.26587
   y( 4) = (y( 4)-y( 3))/(x( 4)-x( 2))
                                    y( 4) =    -.18260
```

```
   y( 3) = (y( 3)-y( 2))/(x( 3)-x( 1))
                                        y( 3) =    -.09293

j = 3
   y(10) = (y(10)-y( 9))/(x(10)-x( 7))
                                        y(10) =    -.02718
   y( 9) = (y( 9)-y( 8))/(x( 9)-x( 6))
                                        y( 9) =    -.05722
   y( 8) = (y( 8)-y( 7))/(x( 8)-x( 5))
                                        y( 8) =    -.08525
   y( 7) = (y( 7)-y( 6))/(x( 7)-x( 4))
                                        y( 7) =    -.11029
   y( 6) = (y( 6)-y( 5))/(x( 6)-x( 3))
                                        y( 6) =    -.13147
   y( 5) = (y( 5)-y( 4))/(x( 5)-x( 2))
                                        y( 5) =    -.14803
   y( 4) = (y( 4)-y( 3))/(x( 4)-x( 1))
                                        y( 4) =    -.15942

j = 4
   y(10) = (y(10)-y( 9))/(x(10)-x( 6))
                                        y(10) =     .04005
   y( 9) = (y( 9)-y( 8))/(x( 9)-x( 5))
                                        y( 9) =     .03737
   y( 8) = (y( 8)-y( 7))/(x( 8)-x( 4))
                                        y( 8) =     .03339
   y( 7) = (y( 7)-y( 6))/(x( 7)-x( 3))
                                        y( 7) =     .02824
   y( 6) = (y( 6)-y( 5))/(x( 6)-x( 2))
                                        y( 6) =     .02209
   y( 5) = (y( 5)-y( 4))/(x( 5)-x( 1))
                                        y( 5) =     .01518

j = 5
   y(10) = (y(10)-y( 9))/(x(10)-x( 5))
                                        y(10) =     .00286
   y( 9) = (y( 9)-y( 8))/(x( 9)-x( 4))
                                        y( 9) =     .00424
   y( 8) = (y( 8)-y( 7))/(x( 8)-x( 3))
                                        y( 8) =     .00550
   y( 7) = (y( 7)-y( 6))/(x( 7)-x( 2))
                                        y( 7) =     .00656
   y( 6) = (y( 6)-y( 5))/(x( 6)-x( 1))
                                        y( 6) =     .00737

j = 6
   y(10) = (y(10)-y( 9))/(x(10)-x( 4))
                                        y(10) =    -.00123
   y( 9) = (y( 9)-y( 8))/(x( 9)-x( 3))
                                        y( 9) =    -.00111
   y( 8) = (y( 8)-y( 7))/(x( 8)-x( 2))
                                        y( 8) =    -.00095
   y( 7) = (y( 7)-y( 6))/(x( 7)-x( 1))
                                        y( 7) =    -.00072

j = 7
   y(10) = (y(10)-y( 9))/(x(10)-x( 3))
                                        y(10) =    -.00009
   y( 9) = (y( 9)-y( 8))/(x( 9)-x( 2))
                                        y( 9) =    -.00012
```

```
eval =    -.00059

eval =     .00761*( t =     .52359-x( 6) =     .93750 )+y( 6) =     .00737

eval =     .00761

eval =     .01345*( t =     .52359-x( 5) =     .75000 )+y( 5) =     .01518

eval =     .01345

eval =    -.15994*( t =     .52359-x( 4) =     .56250 )+y( 4) =    -.15942

eval =    -.15994

eval =    -.11669*( t =     .52359-x( 3) =     .37500 )+y( 3) =    -.09293

eval =    -.11669

eval =     .95493*( t =     .52359-x( 2) =     .18750 )+y( 2) =     .99415

eval =     .95493

eval =     .49999*( t =     .52359-x( 1) =     .00000 )+y( 1) =     .00000
  Resultados de Pi/2  y Pi/6 respectivamente


  1.000000000000000     .499992400000000
```

```
C*************************************************************C
C     programa DVAND.FOR                                      C
C     OBJETIVO : Resuelve el sistema no-confluente            C
C     V(trans)a = f, donde V es una matriz de Vandermonde     C
C     V(alpha[0],...,alpha[n])                                C
C*************************************************************C
      DIMENSION ALPHA(50),A(50),F(50)
      OPEN ( 6,FILE='DVAND.SAL',STATUS='NEW')
      WRITE( *,100)
100   FORMAT( 2X,'Valor de la N = ',I2)
      READ ( *,* ) N
      WRITE(6,200)
200   FORMAT( 3X,'Datos de entrada Alpha, F',/)
      DO 5 I = 1, N
      WRITE( *,250) I
250   FORMAT( 2X,'VALOR DE ALPHA(',I2,') = ',F10.3)
      READ ( *,* ) ALPHA(I)
      WRITE( *,300) I
300   FORMAT( 2X,'              F(',I2,') = ',F10.3,//)
      READ ( *,* ) F(I)
      WRITE( 6,400 ) I, ALPHA(I), I, F(I)
400   FORMAT( 3X,'ALPHA(',I1,')=',F10.3,' F(',I1,')=',F10.3)
5     CONTINUE
      DO 10 K = 1, N
            A(K) = F(K)
10    CONTINUE
      NM1 = N - 1
      DO 15 K = 1, N
            KK = K - 1
            KM1 = K + 1
            DO 15 J = N, KM1, -1
            A(J) = (A(J) - A(J-1)) / (ALPHA(J) - ALPHA(J-KK-1))
15    CONTINUE
      WRITE( 6,500 )
500   FORMAT( //,3X,'Diferencias Divididas a(j)',/ )
      DO 20 I = 1, N
20    WRITE( 6,600 ) I, A(I)
600   FORMAT( 3X,'A(',I2,') =',F10.3 )
      DO 25 K = NM1, 1, -1
            DO 25 J = K, NM1
            A(J) = A(J) - ALPHA(K) * A(J+1)
25    CONTINUE
      WRITE( 6,700 )
700   FORMAT( //,3X,'Coeficientes a(j)',/ )
      WRITE( *,750 )
750   FORMAT( 2X,'EL POLINOMIO P(X) = ' )
      DO 35 I = 1, N
      IM1 = I-1
      IF( IM1 .EQ. 0 ) WRITE(*,760) A(I)
760   FORMAT(23X,F10.3)
      IF( IM1 .EQ. 1 ) WRITE(*,770) A(I)
770   FORMAT(23X,F10.3,' X')
      IF ( IM1 .LE. 1 ) GO TO 30
      WRITE( *,780 ) A(I),IM1
780   FORMAT( 23X,F10.3,' X',I2 )
30    WRITE( 6,800 ) I,A(I)
800   FORMAT( 3X,'A(',I2,') =',F10.3 )
35    CONTINUE
      STOP
      END
```

Datos de entrada Alpha, F

ALPHA(1)=	1.000	F(1)=	10.000
ALPHA(2)=	2.000	F(2)=	26.000
ALPHA(3)=	3.000	F(3)=	58.000
ALPHA(4)=	4.000	F(4)=	112.000

Diferencias Divididas a(j)

A(1) =	10.000
A(2) =	16.000
A(3) =	8.000
A(4) =	1.000

Coeficientes a(j)

A(1) =	4.000
A(2) =	3.000
A(3) =	2.000
A(4) =	1.000

```
C*************************************************************C
C     programa PVAND.FOR                                      C
C     OBJETIVO : Resuelve el sistema no-confluente            C
C     Vx=b, donde V es una matriz de Vandermonde              C
C     V(alpha[0],...,alpha[n])                                C
C*************************************************************C
      DIMENSION ALPHA(50),X(50),B(50)
      OPEN ( 6,FILE='PVAND.SAL',STATUS='NEW')
      WRITE( *,100)
100   FORMAT ( 2X,'VALOR DE N = ',I2 )
      READ ( *,* ) N
      WRITE (6,150)
150   FORMAT( 3X,'Datos de entrada: Alpha, B',/)
      DO 5 I = 1, N
      WRITE( *,200) I
200   FORMAT ( 2X,'VALOR DE X(',I2,') = ',F10.3 )
      READ ( *,* ) ALPHA(I)
      WRITE( *,250) I
250   FORMAT ( 2X,'          B(',I2,') = ',F10.3 )
      READ( *, * ) B(I)
      WRITE (6,300) I,ALPHA(I),I,B(I)
300   FORMAT( 3X,'ALPHA(',I1,')=',F10.3,' B(',I1,')=',F10.3)
5     CONTINUE
      DO 10 K = 1, N
            X(K) = B(K)
10    CONTINUE
      NM1 = N - 1
      DO 20 K = 1, NM1
            KM1 = K + 1
            DO 20 J = N, KM1, -1
            X(J) = X(J) - ALPHA(K) * X(J-1)
20    CONTINUE
      WRITE(6,350)
350   FORMAT(//,3X,'L3(3)L2(2)L1(1) [0,-1,3,35]',/)
      DO 15 I = 1, N
15    WRITE(6,400) I,X(I)
400   FORMAT(3X,'X(',I2,') =',F10.3)
      DO 50 K = NM1, 0, -1
            DO 30 J = K+1, N
            JK1 = J - K - 1
            JK1 = JK1 + 1
            IF ( JK1 .EQ. J ) GO TO 30
C           X(J) = X(J) / (ALPHA(J) - ALPHA(J-K-1))
            X(J) = X(J) / (ALPHA(J) - ALPHA(JK1))
30          CONTINUE
            if ( k .eq. 0 )go to 50
            DO 40 J = K, NM1
            X(J) = X(J) - X(J+1)
40          CONTINUE
50    CONTINUE
      WRITE(6,500)
500   FORMAT(//,3X,'L0(1)TD0-1L1(1)TD1-1L2(1)TD2-1 [0,-1,3,35]',/)
      WRITE(*,550)
550   FORMAT( 2X,'El Polinomio P(x) = ')
      DO 16 I = 1,N
      IM1 = I-1
      IF( IM1 .EQ. 0 ) WRITE( *,600 ) X(I)
600   FORMAT( 23X,F10.3 )
      IF( IM1 .EQ. 1 ) WRITE( *,650 ) X(I)
650   FORMAT( 23X,F10.3,' X')
```

```
      IF ( IN1 .LE. 1 ) GO TO 17
      WRITE( *,700 ) X(I), IN1
700   FORMAT( 23X,F10.3,' X',I2 )
17    WRITE(6,630) I,X(I)
630   FORMAT(3X,'X(',I2,') =',F10.3)
16    CONTINUE
      STOP
      END
```

```
Datos de entrada: Alpha, B

ALPHA(1)=     1.000 B(1)=      .000
ALPHA(2)=     2.000 B(2)=    -1.000
ALPHA(3)=     3.000 B(3)=     3.000
ALPHA(4)=     4.000 B(4)=    35.000


L3(3)L2(2)L1(1) [0,-1,3,35]

X( 1) =       .000
X( 2) =     -1.000
X( 3) =      6.000
X( 4) =      6.000


L0(1)T00-1L1(1)T01-1L2(1)T02-1 [0,-1,3,35]

X( 1) =      3.000
X( 2) =     -4.000
X( 3) =       .000
X( 4) =      1.000
```

```
C*************************************************************
C     programa DUALPRG.FOR                                    *
C     OBJETIVO : Invocar la subrutina del progresivo          *
C                Dual No-confluente                           *
C*************************************************************
      DIMENSION ALPHA(50),C(50),M(50),A(50),F(50)
      OPEN ( 6,FILE='DUALPRG.SAL',STATUS='NEW')
1     WRITE( 6,177 )
177   FORMAT( 3X,'Datos de entrada, Alpha y F',/)
      WRITE(*,100)
100   FORMAT( 2X,'Valor de la N = (valor cero NO)',I2)
      READ ( *, * ) N
      WRITE(*,150) N
150   FORMAT( 2X,'Valor de alpha(',I2,') = ',F10.3)
      READ ( *, * ) ALPHA(N)
      WRITE(*,200) N
200   FORMAT( 2X,'Valor de     F(',I2,') = ',F10.3)
      READ ( *, * ) F(N)
      WRITE( 6,188 ) N,ALPHA(N),N,F(N)
188   FORMAT(3X,'ALPHA(',I1,')=',F10.3,' F(',I1,')=',F10.3)
      CALL DVANPRG(N,ALPHA,C,M,A,F)
      WRITE(*,300)
300   FORMAT( 2X,'Para terminar = 999',/,
     1        2X,'C O N T I N U A R = 0')
      READ ( *, * ) IFIN
      IF ( IFIN .EQ. 999) GO TO 123
      GO TO 1
123   WRITE( *,750 )
750   FORMAT( 2X,'EL POLINOMIO P(X) = ' )
      DO 35 I = 1, N
      IM1 = I-1
      IF( IM1 .EQ. 0 ) WRITE(*,760) A(I)
760   FORMAT(23X,F10.3)
      IF( IM1 .EQ. 1 ) WRITE(*,770) A(I)
770   FORMAT(23X,F10.3,' X')
      IF ( IM1 .LE. 1 ) GO TO 30
      WRITE( *,780 ) A(I),IM1
780   FORMAT( 23X,F10.3,' X',I2 )
30    WRITE( 6,800 ) I,A(I)
800   FORMAT( 3X,'A(',I2,') =',F10.3 )
35    CONTINUE
      STOP
      END
      SUBROUTINE DVANPRG(N,ALPHA,C,M,A,F)
C*************************************************************
C     programa DVANPRG.FOR                                    *
C     OBJETIVO :                                              *
C     Este procedimiento actualiza la soluciÁn al  sistema    *
C     de ecuaciones V(trans)a = f, donde V es noconfluente    *
C     Vandermonde matrix V(alpha[0],...,alpha[n-1]), cuando*  C
C     el valor alpha[n], es agregado. El procedimiento        *
C     debe ser llamado sucesivamente con n=1,2,.....          *
C     El contenido de los arreglos C,M[0:nmax], las cuales    *
C     son usadas como area de almacenamiento no deben de ser  *
C     cambiadas entre invocaciÁnes al sistema.                *
C*************************************************************
      DIMENSION ALPHA(50),C(50),M(50),A(50),F(50)
      C(N) = F(N)
      IF ( N .EQ. 1 ) GO TO 15
      DO 10 J = N-1, 1, -1
```

```
            C(J) = (C(J+1) - C(J)) / (ALPHA(N) - ALPHA(J))
10      CONTINUE
15      CONTINUE
        IF ( N .EQ. 1 ) M(N) = 1
        IF ( N .NE. 1 ) M(N) = 0
        CN = C(1)
        A(N) = CN
              write(*,345) n,cn
              write(6,345) n,cn
345     format(3x,
     1        'Dif.Div. c(',i2,')... =  ',f10.3,'.....')
        IF ( N .EQ. 1 ) RETURN
              NJ = 0
              DO 20 J = N, 2, -1
                    M(J) = M(J) - ALPHA(N-1) * M(J-1)
                    NJ = NJ + 1
                    A(NJ) = A(NJ) + M(J) * CN
20            CONTINUE
        RETURN
        END
```

```
Datos de entrada, Alpha y F

ALPHA(1)=     1.000 F(1)=    10.000
Dif.Div. c( 1)... =       10.000.....
Datos de entrada, Alpha y F

ALPHA(2)=     2.000 F(2)=    26.000
Dif.Div. c( 2)... =       16.000.....
Datos de entrada, Alpha y F

ALPHA(3)=     3.000 F(3)=    58.000
Dif.Div. c( 3)... =        8.000.....
Datos de entrada, Alpha y F

ALPHA(4)=     4.000 F(4)=   112.000
Dif.Div. c( 4)... =        1.000.....
A( 1) =     4.000
A( 2) =     3.000
A( 3) =     2.000
A( 4) =     1.000
```

```
C*************************************************************
C     programa PRIMPRG.FOR                                    *
C     OBJETIVO : Invocar la subrutina del progresivo          *
C                Primal No-confluente                         *
C*************************************************************
      DIMENSION ALPHA(50),D(50),U(50),X(50),B(50)  .
      OPEN ( 6,FILE='PRIMPRG.SAL',STATUS='NEW')
1     WRITE( 6,177 )
177   FORMAT( 3X,'Datos de entrada, Alpha y B',/)
      WRITE(*,100)
100   FORMAT( 2X,'Valor de la N = (valor cero NO)',I2)
      READ ( *, * ) N
      WRITE(*,150) N
150   FORMAT( 2X,'Valor de alpha(',I2,') = ',F10.3)
      READ ( *, * ) ALPHA(N)
      WRITE(*,200) N
200   FORMAT( 2X,'Valor de     B(',I2,') = ',F10.3)
      READ ( *, * ) B(N)
      WRITE( 6,188 ) N,ALPHA(N),N,B(N)
188   FORMAT(20X,'Alpha(',I1,')=',F10.3,' B(',I1,')=',F10.3,/)
      CALL PVANPRG(N,ALPHA,D,U,X,B)
      WRITE(*,300)
300   FORMAT( 2X,'Para terminar = 999',/
     1        2X,'C O N T I N U A R = 0')
      READ ( *, * ) IFIN
      IF ( IFIN .EQ. 999) STOP
      GO TO 1
      END
      SUBROUTINE PVANPRG(N,ALPHA,D,U,X,B)
C*************************************************************
C     programa PVANPRG.FOR                                    *
C     OBJETIVO :                                              *
C     Este procedimiento actualiza la soluciÂn al  sistema    *
C     de ecuaciones Vx = b, donde V es noconfluente           *
C     Vandermonde matrix V(alpha[0],...,alpha[n-1]), cuando*                C
C     el valor alpha[n], es agregado. El procedimiento        *
C     debe ser llamado sucesivamente con n=1,2,.....          *
C     El contenido de los arreglos D,U[0:nmax], las cuales    *
C     son usadas como area de almacenamiento no deben de ser  
C     cambiadas entre invocaciÂnes al sistema.                *
C*************************************************************
      DIMENSION ALPHA(50),D(50),U(50),X(50),B(50)
      D(N) = B(N)
      IF ( N .EQ. 1 ) GO TO 15
      DO 10 J = N-1, 1, -1
           D(J) = D(J+1) - ALPHA(N-J) * D(J)
10    CONTINUE
15    DN = D(1)
      write(6,1234) N,DN
      write(*,1234) N,DN
1234  FORMAT (3X,'Dif.Div.No.'I2,' = ',F10.3)
      U(N) = 1
      IF ( N .EQ. 1 ) GO TO 21
C     FOR   J = 0  STEP 1 UNTIL N-1
      DO 20 J = 1, N-1
           DELTA = ALPHA(N) - ALPHA(J)
           TETA = ALPHA(J) - ALPHA(N)
           U(J) = U(J) * TETA
           U(N) = U(N) * DELTA
           X(J) = X(J) + DN / U(J)
```

```
      write(6,1253) J,X(J)
      write(*,1253) J,X(J)
20          CONTINUE
21    X(N) = DN / U(N)
      write(6,1253) N,X(N)
      write(*,1253) N,X(N)
1253  FORMAT(//,3X,'.........X(',I2,')= ',F10.3)
      RETURN
      END
```

```
Datos de entrada, Alpha y B
                  Alpha(1)=     1.000 B(1)=      .000
Dif.Div.No. 1 =       .000
.........X( 1)=       .000
Datos de entrada, Alpha y B
                  Alpha(2)=     2.000 B(2)=    -1,000
Dif.Div.No. 2 =     -1.000
.........X( 1)=      1.000
.........X( 2)=     -1.000
Datos de entrada, Alpha y B
                  Alpha(3)=     3.000 B(3)=     3.000
Dif.Div.No. 3 =      6.000
.........X( 1)=      4.000
.........X( 2)=     -7.000
.........X( 3)=      3.000
Datos de entrada, Alpha y B
                  Alpha(4)=     4.000 B(4)=    35.000
Dif.Div.No. 4 =      6.000
.........X( 1)=      3.000
.........X( 2)=     -4.000
.........X( 3)=       .000
.........X( 4)=      1.000
```

```
C*************************************************************
C     programa DUALCON.FOR                                    *
C     OBJETIVO : Algoritmo DUAL para calcular las diferen-    *
C                cias divididas y valores finales en el       *
C                caso DUAL CONFLUENTE                         *
C*************************************************************
      DIMENSION F(50),C(50),A(50)
      INTEGER GAMA(9),M(9),GAMAPK,T,S,P,R
      OPEN ( 6,FILE='DUALCON.SAL',STATUS='NEW')
      WRITE( 6,100 )
  100 FORMAT( 3X,'Datos de entrada          ',/)
      WRITE(*,105)
  105 FORMAT( 2X,'Valor de la N = (valor cero NO)',I2)
      READ ( *, * ) N
      WRITE(*,110)
  110 FORMAT( 2X,'Valor de la P = ',I2)
      READ ( *, * ) P
      WRITE(6,111) N,P
  111 FORMAT( 2X,'N = ',I2,'  P = ',I2)
      M(0) = 0
      DO 120 I = 0, P
      WRITE(*,112) I
  112 FORMAT( 2X,'Valor de GAMA (',I2,') = ',I2)
      READ ( *, * ) GAMA(I)
      WRITE(6,113) I,GAMA(I)
  113 FORMAT( 2X,' GAMA (',I2,') = ',I2)
      MG = MG + GAMA(I)
      M(I+1) = MG
      WRITE(6,117) I,M(I)
  117 FORMAT( 2X,' M (',I2,') = ',I2)
  120 CONTINUE
      WRITE(6,117) I,M(I)
      DO 115 II = 0, N
      WRITE(*,114) II
  114 FORMAT( 2X,'Valor de ALPHA(',I2,') = ')
      READ ( *, * ) A(II)
  115 WRITE(6,116) II,A(II)
  116 FORMAT( 2X,' ALPHA (',I2,') = ',F7.3)
C     PASO No. 1 ALGORITMO DUAL ********************************
      DO 135 IJK = 0, N
      WRITE(*,118) IJK
  118 FORMAT(3X,'TECLE EL VALOR F(',I2,') = ')
      READ(*,*) F(IJK)
      WRITE(6,130) IJK,F(IJK)
  130 FORMAT(3X,'      F(',I2,') = ',F7.3)
  135 C(IJK) = F(IJK)
      DO 500 K = 0, N-1
            GAMAPK = GAMA(P) - K
            T = MAX (GAMAPK, 1)
            DO 400 I = P,0,-1
                  IF ( M(I+1) .GT. K+1 ) GO TO 450
                  GO TO 400
  450             IM1 = I - 1
                  S = MAX( M(I),K+1 )
                  IF ( IM1 .LT. 0 ) MGAMA = 0
                  IF ( IM1 .GE. 0 ) MGAMA = GAMA(I-1)
                  R = MAX( MGAMA-K,1 )
                  MIM1 = M(I+1) - 1
                  DO 340 J = MIM1,S,-1
                        MIMT = M(I+1) - T
```

```
                        JKK1 = J - K - 1
                        IF ( J .GT. MINT ) C(J)=C(J)/(K+1)
                        AA = A(J) - A(JKK1)
                        IF ( J .NE. S ) GO TO 310
                        GO TO 320
310                     CJ1 = C(J) - C(J-1)
                        IF ( AA .NE. 0.0 ) C(J) = CJ1 / AA
                        GO TO 330
320                     CJR = C(J) - C(J-R)
                        IF ( AA .NE. 0.0 ) C(J) = CJR / AA
330                     T = R
340                 CONTINUE
400             CONTINUE
500     CONTINUE
        WRITE(6,3333)
3333    FORMAT(//,3X,'LAS DIFERENCIAS DIVIDIDAS C(i)...',//)
        DO 6666 II = 0, N
6666    WRITE(6,7777) II,C(II)
7777    FORMAT( 3X,'C(',I2,') = ',F7.3)
C       PASO No. 2 ALGORITMO DUAL **********************************
C
        DO 5555 L = 0, N
5555    A(L) = C(L)
        DO 2000 K = N-1,0,-1
        DO 2000 J = K,N-1
2000    A(J) = A(J) - A(K) * A(J+1)
        WRITE(6,4444)
4444    FORMAT(//,3X,'FINALMENTE LOS COEFICIENTES A(i)...',//)
        DO 8888 II = 0, N
8888    WRITE(6,9999) II,A(II)
9999    FORMAT( 3X,'A(',I2,') = ',F9.3)
        STOP
        END
```

Datos de entrada

```
N =  5  P =  2
 GAMA ( 0) =  3
 M ( 0) =  0
 GAMA ( 1) =  2
 M ( 1) =  3
 GAMA ( 2) =  1
 M ( 2) =  5
 M ( 3) =  6
 ALPHA ( 0) =   1.000
 ALPHA ( 1) =   1.000
 ALPHA ( 2) =   1.000
 ALPHA ( 3) =   2.000
 ALPHA ( 4) =   2.000
 ALPHA ( 5) =   3.000
       F( 0) =  -1.000
       F( 1) =   4.000
       F( 2) =   5.000
       F( 3) =   6.000
       F( 4) =   7.000
       F( 5) =   8.000
```

LAS DIFERENCIAS DIVIDIDAS C(i)...

```
C( 0) =  -1.000
C( 1) =   4.000
C( 2) =   2.500
C( 3) =    .500
C( 4) =  -3.500
C( 5) =   1.875
```

FINALMENTE LOS COEFICIENTES A(i)...

```
A( 0) =   -17.313
A( 1) = -7384.602
A( 2) = -2160.707
A( 3) =   350.951
A( 4) =    58.521
A( 5) =     1.875
```

```
C*************************************************************
C     programa PRIMCON.FOR                                    *
C     OBJETIVO : Algoritmo PRIMAL                             *
C                                                             *
C                CONFLUENTE                                   *
C*************************************************************
      DIMENSION A(20),C(20),B(20),G(20),X(20)
      INTEGER GAMA(9),M(9),GAMAPK,T,S,P,R
      OPEN ( 6,FILE='PRIMCON.SAL',STATUS='NEW')
      WRITE( 6,100 )
100   FORMAT( 3X,'Datos de entrada           ',/)
      WRITE(*,105)
105   FORMAT( 2X,'Valor de la N = (valor cero NO)',I2)
      READ ( *, * ) N
      WRITE(*,110)
110   FORMAT( 2X,'Valor de la P = ',I2)
      READ ( *, * ) P
      WRITE(6,111) N,P
111   FORMAT( 2X,'N = ',I2,'  P = ',I2)
      M(0) = 0
      DO 115 I = 0, P
      WRITE(*,112) I
112   FORMAT( 2X,'Valor de GAMA (',I2,') = ',I2)
      READ ( *, * ) GAMA(I)
      WRITE(6,113) I,GAMA(I)
113   FORMAT( 2X,' GAMA (',I2,') = ',I2)
      MG = MG + GAMA(I)
      M(I+1) = MG
      WRITE(6,114) I,M(I)
114   FORMAT( 2X,' M (',I2,') = ',I2)
115   CONTINUE
      WRITE(6,114) I,M(I)
      DO 120 II = 0,N
      WRITE(*,116) II
116   FORMAT( 2X,'Valor de ALPHA(',I2,') = ')
      READ ( *,* ) A(II)
120   WRITE (6,121) II, A(II)
121   FORMAT( 3X,'ALPHA (',I2,') = ',F9.3)
      DO 125 III = 0,N
      WRITE(*,122) III
122   FORMAT( 3X,'TECLEE EL VALOR B(',I2,' ) = ',F9.3)
      READ ( *,* ) B(III)
      WRITE (6,123) III, B(III)
123   FORMAT( 3X,'B(',I2,' ) = ',F9.3)
C     PASO No. 1 ALGORITMO PRIMAL ********************************
C
125   G(III) = B(III)
      DO 200 K = 0, N-1
      DO 200 J = N, K+1, -1
200   G(J) = G(J) - A(K) * G(J-1)
      DO 1255 I = 0,N
      WRITE(6,1222) I,G(I)
1255  WRITE(*,1222) I,G(I)
1222  FORMAT( 3X,'VALOR G(',I2,' ) = ',F9.3)
C
C     PASO No. 2 ALGORITMO PRIMAL ********************************
C
      DO 300 IJ = 0,N
300   X(IJ) = G(IJ)
      DO 500 K = N-1, 0, -1
```

```
            DO 400 I = 0, P
                  IF ( M(I+1) .GT. K+1 ) GO TO 450
                  GO TO 400
450               IM1 = I - 1
                  IF ( IM1 .LT. 0 ) MM = 0
                  IF ( IM1 .GE. 0 ) MM = M(I)    .
                  S = MAX(MM,K+1)
                  IF ( IM1 .LT. 0 ) MGAMA = 0
                  IF ( IM1 .GE. 0 ) MGAMA = GAMA(I)
                  T = MAX(MGAMA-K,1)
                  IF ( MM .LE. K+1) R = 1
                  MIM1 = M(I+1) - 1
                  MIMT = M(I+1) - T
                  DO 340 J = S, MIM1
                        JKK1 = J - K - 1
                        AA = A(J) - A(JKK1)
                        IF ( J .GT. MIMT ) GO TO 335
                        IF( AA .NE. 0.0 ) X(J) = X(J)/AA
                        GO TO 340
335                     X(J) = X(J) / (K+1)
340               CONTINUE
                  DO 345 J = S, MIMT
                        IF ( J .NE. S ) GO TO 310
                        X(J-R) = X(J-R) - X(J)
                        GO TO 345
310                     X(J-1) = X(J-1) - X(J)
345               CONTINUE
                  R = T
400         CONTINUE
500   CONTINUE
      WRITE(6,4444)
4444  FORMAT(//,3X,'Los coeficientes X(I)...',/)
      DO 8888 II = 0,N
8888  WRITE(6,9999) II, X(II)
9999  FORMAT( 3X,'X(',I2,')= ',F9.3)
      STOP
      END
```

```
 Datos de entrada

N =  5  P =  2
 GAMA ( 0) =  3
 M ( 0) =  0
 GAMA ( 1) =  2
 M ( 1) =  3
 GAMA ( 2) =  1
 M ( 2) =  5
 M ( 3) =  6
 ALPHA ( 0) =      1.000
 ALPHA ( 1) =      1.000
 ALPHA ( 2) =      1.000
 ALPHA ( 3) =      2.000
 ALPHA ( 4) =      2.000
 ALPHA ( 5) =      3.000
 B( 0 ) =     10.000
 B( 1 ) =     15.000
 B( 2 ) =     25.000
 B( 3 ) =      5.000
 B( 4 ) =     10.000
 B( 5 ) =     15.000
 VALOR G( 0 ) =     10.000
 VALOR G( 1 ) =      5.000
 VALOR G( 2 ) =      5.000
 VALOR G( 3 ) =    -35.000
 VALOR G( 4 ) =    125.000
 VALOR G( 5 ) =   -385.000


 Los coeficientes X(i)...

 X( 0)=   362.500
 X( 1)= -1476.250
 X( 2)=  2351.875
 X( 3)= -1180.000
 X( 4)=   510.000
 X( 5)=   -48.125
```

Bibliografía

1. C. Ballester & Pereyra, (1967) "On the construction of discrete aproximations to linear differential expressions", *Math. Comp.*, v. 21, 1967, pp.297-302.MR 37 # 3751.

2. A. Björck and V. Pereyra (1970) "Solution of Vandermonde Systems of Equations", *Mathematics of Computation 24* , 893-903.

3. G. Galimberti and V. Pereyra (1971) "Solving Confluent Vandermonde Systems of Hermite Type",*Numer. Math. 18,* 44-60.

4. A. Björck and T.Elfving (1973). "Algorithms for Confluent Vandermonde Systems", *Numer. Math. 21,* 130-37.

5. W. Gautschi (1975b). "Optimally Conditioned Vandermonde Matrices" *Numer. Math. 24,* 1-12.

6. G. Galimberti and V. Pereyra (1970). "Numerical Differentiation and the Solution of Multidimensional Vandermonde Systems", *Math. Comp. 24,*357-64.

7. G. H. Golub and C. F. Van Loan, John Hopkins, *Matrix Computations.* Univ. press, Baltimore, Md,1983.

8. J. F. Traub, *Iterative Methods for the solution of Equations,* Prentice-Hall Series in Automatic computation, Prentice-Hall, Englewood Cliffs,N.J.,1964.

9. S. D. Conte and C. de Boor, *Elementary Numerical Analisis: An Algorithmic Approach,* 3rd ed., Mc Graw-Hill, New York, 1980.

10. Thomas R.Mc.Calla. *Introduction to Numerical Methods and Fortran Programming.* John Wiley & Sons. Inc.,New York,1967.

11. Carl-Erik Fröberg. *Introduction to Numerical Analysis,* 2nd ed., Addison-Wesley Publishing Co.,1969.

12. Philip J. Davis. *Interpolation & Aproximation.* Dover Pub. Inc., New York,1975.

13. J. Stoer, R. Bulirsch. *Introduction to Numerical Analysis.* New York: Springer-Verlag,1980.

14. Richard L.Burden, J.Douglas Faire, *Análisis Numérico,* Grupo Editorial Iberoamérica, 1985 Wadsworth Inc.,Belmont California.

15. Gilbert Strang, *Algebra Lineal y sus aplicaciones,* Fondo Educativo Interamericano, Wilington, Delaware U.S.A., 1982.

16. Seymour Lipschutz, *Algebra Lineal,* McGraw-Hill / Interamericana de España, S.A., 2a. edición, 1992.