Universidad Tecnológica de la Mixteca

OBTENCIÓN DE LA FORMA DE UNA SUPERFICIE ASFÉRICA A PARTIR DE RADIOS DE CURVATURA LOCALES UTILIZANDO ALGORITMOS GENÉTICOS

Tesis que para obtener el Título de:

Ingeniero en Computación

Presenta:

Ignacio Gabriel López Licona

Supervisada por:

M.C. Agustín Santiago A.

Huajuapan de León, Oaxaca, Febrero de 2000

I. I. M. 10620

Capítulo III Algoritmos Genéticos

Capítulo IV Implementación del Algoritmo

Capítulo V Pruebas y Resultados

Agradecimientos

A Dios por darme todo lo que tengo y permitirme vivir esta vida...

A mis padres, Ignacio y Ma. Antonieta por su apoyo incondicional para mi formación profesional, por sus enseñanzas, confianza y su gran amor...

A mis hermanas, Raquel y Alejandra por estar siempre apoyándome...

A mis abuelos, Alvaro y Bertha, por estar siempre al pendiente de mi persona y por su gran cariño...

A mis abuelitos, Nacho† y Socorro†, por sus bendiciones desde el cielo...

A mi tía Gabriela, por su ejemplo de superación...

A todos aquellos que de una manera u otra estuvieron al pendiente de la realización de este trabajo...

Un agradecimiento muy especial a mi asesor, M.C. Agustín Santiago A., por su paciencia, dedicación y apoyo...

U. A. N. L. 10620

Introducción

En el avance de la ciencia y la tecnología, los sistemas ópticos que forman parte de los instrumentos, juegan un papel fundamental para el crecimiento de la investigación y el descubrimiento científico. Sin el desarrollo de estos sistemas ópticos, muchos de los avances de la ciencia no hubieran sido posibles, imaginemos por ejemplo, a la biología sin un microscopio, tal vez nunca hubiéramos conocido la estructura de nuestras células, a la astronomía sin un telescopio, no sabríamos que hay en el espacio exterior más haya de nuestros ojos, a la medicina sin una cámara de vídeo, una endoscopia sería un sueño, y así podemos seguir mencionando más y más ejemplos de la importancia que han tenido los sistemas ópticos en el transcurso de la evolución científica y tecnológica.

Desde los inicios de la construcción de sistemas ópticos, estos se han fabricado con superficies de forma esférica, ya que estas son más sencillas de producir, con la desventaja de que se requieren de muchas componentes ópticas para lograr una buena calidad en la imagen.

Actualmente, por la necesidad de contar con instrumentos ópticos de alta calidad que satisfagan aplicaciones cada vez más sofisticadas y que no pueden ser resueltas por los instrumentos convencionales, ha sido necesario introducir superficies ópticas de forma *asférica* como componentes de los sistemas ópticos, logrando con estas que los instrumentos ópticos sean más compactos y eficientes, además de ocupar menos espacio y ser mucho más ligeros.

La base para el desarrollo de un sistema óptico radica en un buen diseño, una vez obtenido este, el instrumento óptico debe construirse cumpliendo con cada una de las especificaciones señaladas por el diseñador, como son: (formas de las superficies, tipo de material que será utilizado, grosor, separaciones, tamaño, etc.), ya que si estas varían, el funcionamiento del instrumento no será el deseado.

Durante el proceso de construcción de un sistema óptico, es muy importante realizar una **etapa de pruebas** ópticas, la cuál se encargará de verificar que las componentes del instrumento (por separado y como un solo elemento), cumplan con las especificaciones dadas previamente por el diseñador.

Los métodos de prueba ópticos que existen actualmente, en su mayoría, funcionan bien para superficies esféricas y con forma cóncava, debido a que se hace incidir un haz de luz sobre su superficie y esta hace converger el haz, permitiendo con esto obtener información sobre la superficie en alguna región pequeña.

El problema surge cuando se requiere probar para superficies asféricas y de forma convexa, los métodos tradicionales se vuelven obsoletos, además de que los costos de implementación son muy altos, debido a que las componentes ópticas auxiliares que se requieren deben ser de tamaño más grande a la superficie que se desea probar.

Este problema a dado pie al nacimiento de nuevas técnicas o bien a la modificación de los métodos ya existentes, uno de estos métodos consiste en medir radios de curvatura en diferentes partes de la superficie, y a partir de ellos obtener la forma de la superficie. El éxito de esta técnica radica en optimizar la forma de la superficie que mejor se ajusta a dichos puntos.

En el presente trabajo tiene como objetivo implementar un algoritmo de optimización para encontrar la forma de una superficie asférica a partir de radios de curvatura locales, dicho algoritmo tendrá como objetivo ser una herramienta de apoyo para la etapa de pruebas en sistemas ópticos, en otras palabras se realizará un algoritmo que permita obtener la forma de una superficie que mejor se ajuste a una distribución de puntos (x, y, φ), donde φ es el radio de curvatura local.

La superficie encontrada describirá analíticamente dicha distribución de puntos, todo esto cae dentro de un problema de optimización por lo cual se propone implantar un algoritmo que trabaje con la técnica de **Algoritmos Genéticos** para la obtención de la forma citada.

El contenido del trabajo se presenta como sigue:

En el capítulo uno se hará una revisión a los métodos de optimización que se han utilizado tradicionalmente en el diseño de sistemas ópticos, analizando sus ventajas y desventajas.

En el capítulo dos, se presentarán los tipos de superficies más comúnmente usadas en óptica.

En el capítulo tres, se estudiarán los algoritmos genéticos, se darán sus fundamentos, se explicará su funcionamiento, así como sus ventajas y desventajas, se explicará porque es un método de optimización.

En el capítulo cuatro, se presenta la forma en que se implantó el algoritmo genético. Finalmente en el capítulo cinco, se darán unos ejemplos y resultados obtenidos por el algoritmo.

CAPÍTULO I

MÉTODOS DE OPTIMIZACIÓN

Introducción.

En el diseño óptico, el espacio de solución de un sistema óptico es un espacio multidimensional de N variables independientes las cuales representan los parámetros del diseño tales como rádios de curvatura, espacios de aire, espesores de vidrio, índices de refracción, etc., para el cual se define una función de mérito la cual se evalúa durante el proceso de optimización.

De tal forma que un punto en el espacio multidimensional, es una solución o configuración específica del sistema óptico y la función de mérito evaluada en tal punto es una medida de la calidad del sistema óptico.

Una vez diseñado el sistema óptico se prosigue a construirlo, durante el proceso de construcción del sistema óptico, es muy importante contar con una etapa de ***pruebas ópticas***, la cual tiene el objetivo de verificar que todas las componentes del instrumento, cumplan con las especificaciones dadas previamente por el diseñador.

Para el caso de superficies ópticas convexas, existen métodos de prueba que realizan mediciones locales y a partir de ellos conocer la forma de las superficies. El éxito en este tipo de pruebas radica en contar con un buen método de optimización.

Los métodos convencionales de optimización empleados en óptica, trabajan satisfactoriamente y convergen rápidamente a una buena solución sólo cuando un buen punto de partida ha sido seleccionado. En caso contrario no se alcanza una buena solución, como se verá más adelante.

En el presente capítulo, se revisan algunos de estos métodos de optimización.

1.1 La función de mérito.

Para conocer la forma de la superficie, se tiene como premisa fundamental, que la calidad de la forma de la superficie se pueda especificar con un solo número real dado por una función llamada de mérito ϕ.

Por tanto, puede decirse que la obtención de la forma de la superficie, consiste en encontrar el extremo de una función (máximo o mínimo dependiendo de la forma exacta de la definición de ϕ) seleccionada adecuadamente y sujeta a restricciones para evitar la obtención de superficies poco prácticas o bien inservibles.

Para el caso que se desea resolver, la función de mérito depende primordialmente de los parámetros a optimizar (C, K) de la superficie, donde C es la curvatura y K es la constante de conicidad. En los siguientes apartados se define dicha función de mérito y se analiza para cada uno de los métodos presentados.

1.2 Métodos convencionales de optimización.

La función de mérito debe ser continua y diferenciable, comúnmente se define como; [1]

$$\phi = \sum_{i=1}^{M} f_i^2, \qquad (1.1)$$

donde las funciones f_i dependen de los parámetros a optimizar.

Las funciones f_i miden las desviaciones de los radios de curvatura del sistema óptico y tienen la forma general;

$$f_i = (R_{ie} - R_{iT})^2, \qquad (1.2)$$

donde R_{ie} son los radios de curvatura medidos experimentalmente y permanecen constantes y R_{iT} están definidos por; [2]

$$R_{iT} = \frac{1}{C}\left(1 - KC^2S^2\right)^{1/2}, \qquad (1.3)$$

donde C en la curvatura paraxial, $S^2 = x^2 + y^2$, K es la constante de conicidad y R_{iT} son los valores que encuentra el algoritmo y se comparan con R_{ie}.

La función mérito ϕ también puede ser definida en notación matricial; [3]

$$\phi = F^T F, \qquad (1.4)$$

donde F es un vector columna cuyas componente son las f_i, y F^T es su transpuesto.

Si se realiza un desarrollo de cada función f_i en una serie de Taylor, y se corta la serie después de los términos con primeras derivadas, entonces se tiene; [3]

$$\phi = \sum_{i=1}^{M}\left[f_{0i} + \sum_{j=1}^{N} \frac{\partial f_i}{\partial x_j}\left(x_j - x_{0j}\right)\right]^2, \qquad (1.5)$$

donde f_{0i} es el valor de f_i en $\mathbf{X}_0$. $\mathbf{X}_0$ es un punto en el espacio de solución formado por los valores x_{0j} que representan el punto de partida al comenzar el proceso de optimización.

desarrollando el binomio al cuadrado de la ecuación (1.5), se tiene; [3]

$$\phi = \sum_{i=1}^{M} f_{0i}^2 + 2\sum_{i=1}^{M}\left[f_{0i}\sum_{j=1}^{N} a_{ij}\left(x_j - x_{oj}\right)\right] + \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=1}^{N} a_{ij}a_{ik}\left(x_j - x_{0j}\right)\left(x_k - x_{0k}\right), \qquad (1.6)$$

donde;

$$a_{ij} = \frac{\partial f_i}{\partial x_i}. \qquad (1.7)$$

En la ecuación (1.6) el primer término es constante y puede despreciarse porque no influye en la topografía del espacio de solución. El segundo y tercer término pueden combinarse cambiando el origen de x_j y haciendo una rotación de ejes de tal forma que ϕ pueda expresarse como una forma cuadratica definida positiva [3] ($a^2_{ij} \geq 0$ para todo i, j).

En notación matricial se tiene;

$$\phi = \left(X - X_0\right)^T A^T A\left(X - X_0\right), \qquad (1.8)$$

donde A es una matriz de M x N con a_{ij} elementos.

Las curvas de nivel de la función de mérito ϕ son elipsoides en un espacio multidimensional de N variables y sus excentricidades están determinadas por los valores de [3] a_{ij}. Las longitudes de los N ejes principales de los elipsoides están dados por los eigenvalores de la matriz A^TA.

A continuación se describen los métodos más comunes que se utilizan para optimizar un sistema.

1.2.1 Métodos de búsqueda directa.

En estos métodos, la computadora simplemente examina secuencialmente un conjunto de soluciones de prueba y cada solución es comparada con el conjunto de soluciones anteriores.

Aunque estos métodos se utilizan con mucho éxito en varios campos, en el caso del diseño óptico y pruebas, poco éxito han tenido, ya que el tiempo de computo necesario es muy grande, aún con equipos de computo actuales.[3]

1.2.2 Métodos Matemáticos.

Estos métodos se fundamentan en el cálculo diferencial para establecer estrategias eficientes con el objetivo de mejorar sistemas a partir de un sistema inicial.

Todos estos métodos utilizan solamente las primeras derivadas, porque el trabajo involucrado y el tiempo de computo se hace muy grande en el cálculo de las derivadas de orden superior. [3]

De la ecuación (1.5) se tiene;

$$f_i = f_{0i} + \sum_{j=1}^{N} a_{ij}\left(x_j - x_{0j}\right), \quad \text{para } i = 1, 2, 3, \ldots, M \tag{1.9}$$

su equivalente en notación matricial;

$$F = F_0 + A\left(X - X_0\right). \tag{1.10}$$

Si se considera a $f_i = 0$ para todo i, se obtiene un sistema de ecuaciones lineales simultáneas en $x_j - x_{0j}$, y al resolverlo se obtienen los cambios de las N variables que dan origen a un sistema con desviaciones igual a cero. Es decir se obtiene la ecuación;

$$A(X - X_0) = -F_0. \tag{1.11}$$

El método anterior tiene la desventaja de que la matriz $\boldsymbol{A}$ debe ser cuadrada. Es decir, debe haber tantas variables como ecuaciones, lo cual frecuentemente no sucede en el diseño óptico. Además como la matriz $\boldsymbol{A}$ es singular [3], las soluciones de la ecuación (1.9) son muy grandes en magnitud y por tanto las no linealidades de $\boldsymbol{F}$ afectan seriamente la solución que en muchos casos resulta ser peor respecto a la anterior.

1.2.3 Mínimos Cuadrados.

Otra propuesta diferente a la de la sección anterior, es minimizar los valores de las f_i en lugar de igualarlas a cero, para esto se deriva a ϕ con respecto a cada x_j, de la ecuación (1.1) y con ayuda de (1.7) se tiene;

$$\frac{\partial \phi}{\partial x_k} = \sum_{i=1}^{M} 2 f_i a_{ik}, \tag{1.12}$$

para $k = 1,2, \ldots N$

Sustituyendo f_i de la ecuación (1.9) e igualando a cero en (1.12), se obtienen las siguientes N ecuaciones en $x_j - x_{0j}$;

$$\sum_{i=1}^{M} a_{ik} f_{0i} + \sum_{i=1}^{M} \sum_{j=1}^{N} a_{ij} a_{ik} (x_j - x_{0j}) = 0, \tag{1.13}$$

o en notación matricial;

$$A^T A(X - X_0) + A^T F_0 = 0. \tag{1.14}$$

Las ecuaciones (1.13) o (1.14) son las ecuaciones clásicas de mínimos cuadrados, estas no necesitan que el número de incógnitas sea igual al de las variables.

La solución de la ecuación (1.14) es;

$$X - X_0 = -\left(A^T A\right)^{-1} A^T F_0. \tag{1.15}$$

Donde es claro que pequeños errores en el cálculo de las derivadas o los errores de redondeo al resolver las ecuaciones tendrán un efecto importante sobre la solución que se obtenga, pero aunque la exactitud fuera muy grande, la no linealidad de F hace que la solución dada en (1.15) sea un óptimo solo en el caso lineal, y por lo tanto, es frecuente que la solución conduzca a un sistema óptico peor que el inicial.

1.2.4 Mínimos cuadrados amortiguados.

Una forma de resolver las dificultades antes mencionadas es limitar los cambios en los parámetros de tal forma que se mantenga una aceptable correlación entre las mejoras pronosticadas y las mejoras reales.

Para implementar lo mencionado, Wynne [4] remplazó la función de mérito ϕ por;

$$\phi = \sum_{i=1}^{M} f_i^{\,2} + P^2 \sum_{j=1}^{N} \left(x_j - x_{0j}\right)^2, \tag{1.16}$$

donde P es un escalar, de tal forma que las ecuaciones (1.13) y (1.14) se trasforman en;

$$\sum_{i=1}^{M} a_{ik} f_{0i} + \sum_{i=1}^{M} \sum_{j=1}^{N} \left[a_{ij} a_{ik} \left(x_j - x_{0j}\right) + P^2 \left(x_j - x_{0j}\right)\right] = 0, \tag{1.17}$$

para k = 1,2, ...N

y en notación matricial;

$$\left(A^T A + P^2 I\right)\left(X - X_0\right) = -A^T F_0, \tag{1.18}$$

donde $\boldsymbol{I}$ es la matriz unitaria de orden N y $\boldsymbol{P}$ es un escalar que determina el tamaño del paso. En la practica el valor de $\boldsymbol{P}$ esta determinado por la no linealidad del sistema óptico. Por ejemplo, si la discrepancia entre los valores reales de las f_i y los valores pronosticados es muy grande, entonces $\boldsymbol{P}$ se incrementa hasta que algún nivel de coincidencia se alcance. Alternativamente, los valores de ϕ pueden calcularse para distintos valores de $\boldsymbol{P}$ y ajustando una curva se puede encontrar el valor óptimo de $\boldsymbol{P}$.

El procedimiento anterior de amortiguamiento es incorrecto, debido a que trata a todas las variables de la misma manera, siendo que la sensibilidad de la solución a errores de redondeo o a errores en las derivadas es diferente de variable a variable.

Al procedimiento mencionado se le conoce como *amortiguamiento aditivo*[5] y una generalización de este se obtiene cuando se usa como función de mérito a;

$$\phi = \sum_{i=1}^{M} f_i^2 + P^2 \sum_{j=1}^{N} q_j^2 \left(x_j - x_{0j}\right)^2, \tag{1.19}$$

propuesta por Meiron [3], donde los factores q_j se calculan como;

$$q_j^2 = \sum_{i=1}^{M} a_{ij}^2. \tag{1.20}$$

De tal forma que las variables que ocasionan un mayor cambio en ϕ son altamente amortiguadas. La ecuación (1.18) se convierte en;

$$\left(A^T A + P^2 Q\right)\left(X - X_0\right) = -A^T F_0, \tag{1.21}$$

donde Q es una matriz diagonal cuyos elementos son q^2_j, y como los elementos de la diagonal de A^TA son iguales a;

$$\sum_{i=1}^{M} a_{ij}^2. \qquad (1.22)$$

Así las ecuaciones amortiguadas se obtienen de las ecuaciones de mínimos cuadrados simplemente multiplicado los términos de la diagonal por $I+P^2$, dando lugar a lo que se conoce como *amortiguamiento multiplicativo.*[5]

Este último procedimiento tiene poca justificación teórica, y q_j debe ser determinado por la magnitud de las segundas derivadas y no existe ninguna razón para suponer que una primera derivada con un valor grande implique una segunda derivada también con un valor grande, sin embargo, a pesar de la falla de sustento teórico ambos amortiguamientos, el aditivo y el multiplicativo, se han usado con un éxito considerable dentro del diseño óptico.

1.2.5 Método del Gradiente.

Buchele [3], ha sugerido un método diferente al de amortiguar las ecuaciones de mínimos cuadrados usando una justificación teórica más concreta.

Si se tiene un sistema inicial X_0 y se quiere predecir pasos $X - X_0$ que den como resultado un sistema X para el cual ϕ tiene un mínimo, entonces el gradiente de ϕ debe ser cero y por lo tanto, se puede usar el método de Newton – Raphson, el cual es muy útil para encontrar los ceros de una función. En este caso la función de la cual se esta buscando los ceros es $\nabla\phi$ y usando la definición de ϕ dada en la ecuación (1.1) junto con el método de Newton – Raphson se obtiene; [3]

$$\sum_{j=1}^{N} \frac{\partial(\nabla\phi_k)}{\partial x_j}\left(x_j - x_{0j}\right) = -\nabla\phi_k, \qquad (1.23)$$

donde $\nabla\phi_i$ es la i-ésima componente de $\nabla\phi$.

Desarrollando los gradientes y simplificando resulta;

$$\sum_{j=1}^{N}\sum_{i=1}^{M}\left[a_{ij}a_{ik}+f_{0i}\frac{\partial^2 f_i}{\partial x_i \partial x_k}\right]\left(x_j-x_{0j}\right)=-\sum_{i=1}^{M}f_{0i}a_{ik}, \qquad (1.24)$$

En la ecuación (1.24) si se desprecian los términos que incluyen a las segundas derivadas, se obtienen las ecuaciones clásicas de mínimos cuadrados, se observa que las segundas derivadas proporcionan el amortiguamiento necesario para corregir las oscilaciones que se tienen al optimizar con mínimos cuadrados, sin embargo, la cantidad de computo necesario para evaluar N^2 segundas derivadas es muy grande, y Buchele aproxima las ecuaciones (1.24) despreciando todas las derivadas cruzadas de tal forma que solo se tengan que evaluar N segundas derivadas. Así el termino de amortiguamiento será;

$$\sum_{j=1}^{N}\sum_{i=1}^{M}f_{0i}\frac{\partial^2 f_i}{\partial x_j^2}\left(x_j-x_{0j}\right)=F_0^T B\left(X-X_0\right), \qquad (1.25)$$

donde $\boldsymbol{B}$ es una matriz con elementos dados por;

$$b_{ij}=\frac{\partial^2 f_i}{\partial x_j^2}, \qquad (1.26)$$

en notación matricial el conjunto de ecuaciones queda definido por;

$$\left(A^T A+F_0^T BI\right)\left(X-X_0\right)=-A^T F_0. \qquad (1.27)$$

Buchele [3] menciona que para un sistema con dos variables el amortiguamiento con segundas derivadas es más eficiente que los amortiguamientos aditivo y multiplicativo.

1.2.6 Métodos Descendentes.

Los métodos de optimización descendentes fueron descritos por Feder [3] y su filosofía es de predecir alguna dirección D_0 tal que ϕ disminuya a lo largo de esta dirección. Es decir, desde un sistema inicial X_0 se hace un desplazamiento hasta el sistema X_1 de tal forma que;

$$X_1 - X_0 = hD_0, \tag{1.28}$$

donde h es un escalar y se calcula para que se obtenga el mínimo de ϕ a lo largo de D_0. Para el caso de funciones lineales [3], h puede calcularse multiplicando la ecuación (1.28) por la matriz A;

$$A(X_1 - X_0) = hAD_0, \tag{1.29}$$

y como F es lineal en X se tiene;

$$F_1 - F_0 = A(X_1 - X_0), \tag{1.30}$$

donde F_1 y F_0 son los valores de F en X_1 y X_0 respectivamente. Sustituyendo (1.30) en (1.29) se tiene;

$$F_1 - F_0 = hAD_0, \tag{1.31}$$

ahora multiplicando esta ecuación por A^T;

$$A^T F_1 - A^T F_0 = hA^T AD_0, \tag{1.32}$$

considerando que $G_0 = \nabla\phi$ y $A^T F_0 = \frac{1}{2} \nabla\phi$ se tiene que;

$$1/2(G_1 - G_0) = hA^T AD_0. \tag{1.33}$$

Donde G_1 es el gradiente de ϕ en X_1 multiplicando por D_o^T, se tiene;

$$1/2\left(D_0^T G_1 - D_0^T G_0\right) = hD_0^T A^T AD_0. \qquad (1.34)$$

Como h debe seleccionarse de tal forma que ϕ sea un mínimo a lo largo de D_0 en X_1, entonces $D_0^T G_1 = 0$. Y despejando h se tiene;

$$h = -\frac{D_0^T G_0}{2D_0^T A^T AD_0}. \qquad (1.35)$$

1.2.7 La métrica

La selección más razonable para D_0 es la dirección de más decrecimiento con respecto a la distancia. Sin embargo, en el espacio de solución la distancia no se encuentra bien definida porque las variables de dicho espacio no tienen las mismas dimensiones, de ahí la necesidad de definir la distancia S, desde el origen hasta un punto X. La definición de S queda expresada como; (3)

$$S^2 = X^T BX, \qquad (1.36)$$

donde B es una matriz simétrica definida positiva.

Cuando S se define como en (1.36) se dice que se mide respecto a la métrica B y solo cuando B esta definida tiene sentido mencionar la dirección de más rápido descenso de ϕ. (3)

La elección más natural para la métrica es la del espacio Euclidiano, es decir, cuando B es la matriz unitaria I, de tal manera que el movimiento desde X_0 hasta X_1 sea en la dirección inversa a la del vector gradiente G_0.

De las ecuaciones (1.35) y (1.28) se obtiene; (3)

$$X_1 - X_0 = -\frac{1}{2}\frac{G_0^T G_0}{G_0^T A^T AG} G_0. \qquad (1.37)$$

En la práctica, el uso de esta métrica conduce a una convergencia muy lenta porque el incremento en la longitud del paso es muy pequeño, pero esto a su vez garantiza que la función es lineal, en el intervalo dado, dando lugar a un proceso de convergencia estable.

Ahora, si se usa como métrica a la matriz A^TA, la dirección D_0 será; [3]

$$D_0 = -\left(A^T A\right)^{-1} G_0, \tag{1.38}$$

y de (1.35) el valor de h estará dado por;

$$h = -\frac{1}{2}\frac{G_0^T\left[\left(A^T A\right)^{-1}\right]^T G_0}{G_0^T\left[\left(A^T A\right)^{-1}\right]^T\left(A^T A\right)\left(A^T A\right)^{-1} G_0} = -\frac{1}{2}, \tag{1.39}$$

sustituyendo (1.38) y (1.39) en (1.28), se obtiene;

$$X_1 - X_0 = -\frac{1}{2}\left(A^T A\right)^{-1} G_0, \tag{1.40}$$

y recordando que $G_0 = 2A^TF_0$, se obtiene la siguiente ecuación;

$$A^T A\left(X_1 - X_0\right) = -A^T F_0. \tag{1.41}$$

Esta última ecuación es idéntica a la ecuación (1.14), y por lo tanto, es la expresión matricial de las ecuaciones de mínimos cuadrados. Entonces, puede considerarse al método de mínimos cuadrados como un método descendente con una métrica A^TA.

De las ecuaciones (1.6) y (1.8) se tiene que;

$$\phi = \phi_0 + \left(X - X_0\right)^T A^T A\left(X - X_0\right). \tag{1.42}$$

Pero la métrica es $A^T A$ y usando la ecuación (1.36) se tiene;

$$\phi - \phi_0 = S^2. \tag{1.43}$$

Donde se obtiene que en un espacio con una métrica $A^T A$, las curvas de nivel de ϕ son hiperesferas concéntricas. El gradiente de ϕ es normal a las esferas en todos los puntos y pasa por el centro de las familias de las esferas. Por lo tanto, el mínimo se encuentra en un solo paso.

El método descendente usando la métrica $A^T A$ da lugar a pasos grandes y a una rápida convergencia, pero por la no linealidad hay problemas de estabilidad en el proceso y resultan oscilaciones invariablemente.

Se puede establecer un compromiso entre la estabilidad de la métrica I y la eficiencia de la métrica $A^T A$, una elección lógica será; [3]

$$A^T A + P^2 I. \tag{1.44}$$

La métrica anterior corresponde claramente al método de mínimos cuadrados amortiguados.

1.2.8 Multiplicadores de Lagrange.

En los procedimientos de optimización que se mencionaron con anterioridad, se trata de minimizar la función de mérito dada en la ecuación (1.1). Sin embargo, en muchos casos no es suficiente tener una minimización sin restricciones, ya que puede desearse que el sistema óptico posea algunas propiedades preasignadas. Es decir, se trata de que algunas ecuaciones se resuelvan exactamente al mismo tiempo que la función de mérito se minimiza. Para esto, se añade la condición de que la siguiente ecuación se debe satisfacer exactamente; [3]

$$B^T (X - X_0) + D = 0, \tag{1.45}$$

donde $\boldsymbol{B}$ es una matriz de P x N elementos, los cuales son;

$$b_{kj} = \frac{\partial q_k}{\partial x_j}, \tag{1.46}$$

donde q_k es un conjunto de funciones que tienen valores dados por c_k que definen las propiedades preasignadas en q_k y en c_k, k=1, 2, ..., P, donde P < N.

Cada una de las restricciones representa un plano N-dimensional, cuyo vector normal es; [3]

$$V_k = \begin{bmatrix} b_{k1} \\ \vdots \\ b_{kN} \end{bmatrix}. \tag{1.47}$$

La intersección de todos los planos es una superficie (N-P) - dimensional sobre la cual debe encontrarse el mínimo de la función de mérito que satisfaga las restricciones. Como el mínimo es un punto estacionario de ϕ, el gradiente de ϕ no tiene componente sobre esta superficie, así que puede expresarse como una combinación lineal de todos los vectores lineales a la superficie; [3]

$$\nabla_\phi = 2\sum_{k=1}^{P} \lambda_k V_k, \tag{1.48}$$

donde las λ_k forman un conjunto de multiplicadores escalares. Considerando Λ como un vector columna con elementos λ_k, la ecuación (1.48) se puede expresar en notación matricial como;

$$\nabla_\phi = 2B^T \Lambda, \tag{1.49}$$

donde B^T es el transpuesto de V_K.

Recordando que $\nabla_\phi = 2A^T F$ e igualando con (1.49) se tiene;

$$A^T F = B^T \Lambda, \tag{1.50}$$

de aquí;

$$A^T A(X - X_0) - B^T \Lambda = -A^T F, \tag{1.51}$$

donde los escalares λ_k son incógnitas en la ecuación y se conocen como los multiplicadores de Lagrange.[3]

Las ecuaciones (1.45) y (1.51) forman un conjunto de N+P ecuaciones con N+P incógnitas, por lo tanto, considerar el conjunto de restricciones significa resolver un conjunto de ecuaciones, y como las λ_k no interesan, pueden permanecer desconocidas.

1.3 Conclusiones.

Se presentaron de manera resumida los métodos de mínimos cuadrados, mínimos cuadrados amortiguados, gradiente y los métodos descendientes, así como el de Lagrange.

Se mostró que los métodos de mínimos cuadrados amortiguados y mínimos cuadrados son métodos descendentes con métricas distintas, también se presentó que sus deficiencias al trabajar con funciones no lineales se debe al hecho de no incluir derivadas de orden superior.

En suma, todos los métodos antes presentados, necesitan un punto inicial, a partir del cual comienza la búsqueda del óptimo, lo cual conduce irremediablemente a óptimos locales y a una dependencia muy estrecha del punto inicial.

CAPÍTULO II

SUPERFICIES

Introducción.

En el presente capítulo se define lo que es un Lugar Geométrico, tipo de superficies comúnmente empleadas en óptica[6], las superficies cuádricas, de revolución, se define que es una superficie óptica, así como las superficies cónicas, por último se presentan la superficies asféricas.

2.1 Superficies Notables como Lugares Geométricos.

Se define como Lugar Geométrico, al conjunto de todos los puntos del espacio que gozan de una misma propiedad, propiedad expresada en la definición del lugar geométrico.[7]

La expresión matemática que representa a un lugar geométrico, se llama Ecuación o Ecuaciones del lugar.

Existen dos métodos analíticos generales para obtener un lugar geométrico, el **directo** y el **indirecto**.[8]

El **método directo**, consiste en representar por las variables x, y, z, las coordenadas de un punto del lugar y traducir de manera inmediata en forma algebraica, por medio de ecuaciones, la propiedad que gozan todos los puntos del lugar.

Ejemplo 2.1: Lugar geométrico de los puntos del espacio que distan de un punto fijo, $C\ (x_0, y_0, z_0)$, una longitud constante r (esfera)[7]

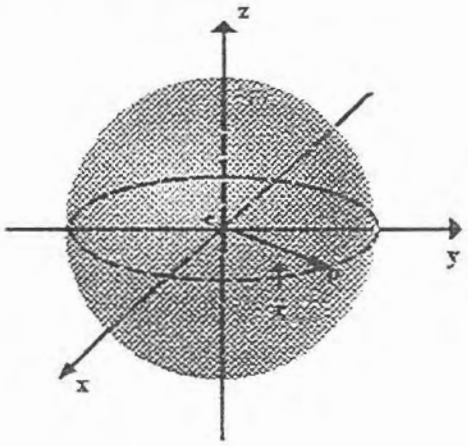


Figura 2.1. Representación de un Lugar Geométrico.

La distancia entre los puntos P y C es igual a r

$$\overline{PC}^2 = r^2 = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 \qquad (2.1)$$

La ecuación (2.1) es la que da el lugar geométrico de una esfera.

En el **método Indirecto**, se consideran las superficies como lugares de líneas (generatrices) obtenidas por intersección de dos superficies variables, que dependen de uno o más parámetros, y que se mueven con respecto a leyes determinadas que condicionan dichos parámetros (cuando hay más de uno).

2.2 Superficies.

Desde el punto de vista de la óptica geométrica, un sistema óptico es un instrumento que permite la manipulación de un "manojo" de rayos con el objetivo casi siempre de formar una imagen, para ello se utilizan lentes, espejos, prismas, etc., cuya función es cambiar la dirección de propagación de los rayos.

Las lentes están formadas por dos o más interfaces refractoras, donde al menos una de estas es curvada. Estas interfaces son secciones de superficies de revolución casi siempre, como cilindros, cónicas, toroides, superficies asféricas, etc.

A continuación se mencionan las características más generales de las mismas.

2.2.1 Cilindros.

En el espacio bidimensional, la gráfica de $x^2 + y^2 = 1$, es una circunferencia con centro en el origen.[20] Sin embargo, en un espacio tridimensional, es posible interpretar la gráfica del conjunto

$$\{(x, y, z) \mid x^2 + y^2 = a, z\} \qquad \text{donde } z \text{ es arbitrario y } a \text{ es constante,} \qquad (2.2)$$

como una superficie, que es el cilindro circular recto, el cual se muestra en la figura 2.2.

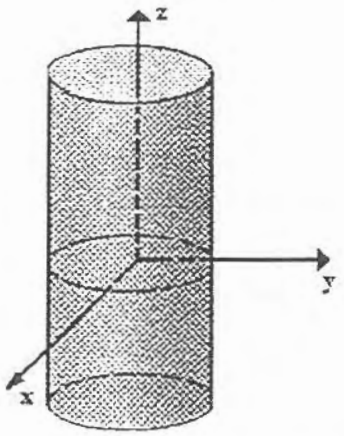


Figura 2.2. Cilindro Circular Recto.

2.2.2 Esferas.

Una esfera es el conjunto de todos los puntos P del espacio tridimensional que equidistan de un punto fijo llamado centro.[9]

Si r denota la distancia fija, o radio de la esfera, y si el centro es $P_1(a, b, c)$, entonces un punto $P(x, y, z)$ se encuentra en la esfera si y solo si $d(P_1, P_2)^2 = r^2$, o sea;

$$(x-a)^2 + (y-b)^2 + (z-c)^2 = r^2. \qquad (2.3)$$

Ejemplo 2.2: La gráfica de $x^2 - y^2 - z^2 = 25$ es una esfera de radio 5 con centro en el origen, ya que a = 0, b = 0, c = 0. La gráfica de la ecuación se presenta en la figura 2.3.

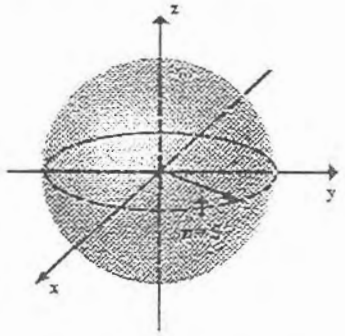


Figura 2.3. Gráfica de la Ecuación; $x^2 + y^2 + z^2 = 25$.

2.2.3 Traza de una Superficie.

La traza de una superficie es una curva formada por la intersección de una superficie y un plano coordenado.[9] En la figura 2.3, la traza de la esfera en el plano xy es la circunferencia $x^2 + y^2 = 25$. En los planos xz y yz, las trazas de la esfera son las circunferencias $x^2 + z^2 = 25$ y $y^2 + z^2 = 25$, respectivamente.

2.3 Superficies Cuádricas.

La ecuación de la esfera (2.3) es solo un caso particular de la ecuación de segundo grado,[9]

$$Ax^2 + By^2 + Cz^2 + Dx + Ey + Fz + G = 0. \qquad (2.4)$$

Cuando A, B y C son diferentes a cero, se dice que la gráfica de una ecuación de la forma (2.4) es una superficie cuádrica.[9]

Otra definición similar, propuesta por Raúl Cárdenas[7], una superficie cuádrica es el lugar geométrico de los puntos del espacio que satisfacen la ecuación algebraica de segundo grado en tres variables;

$$a_{11}x^2 + a_{22}y^2 + a_{33}z^2 + 2a_{12}xy + 2a_{13}xz + 2a_{23}yz + 2a_{14}x + 2a_{24}y + 2a_{34}z + a_{44} = 0. \qquad (2.5)$$

A continuación se consideran cinco superficies cuádricas.

2.3.1 Elipsoide.

La gráfica de cualquier ecuación de la forma:[9]

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1, \quad a, b, c > 0, \tag{2.6}$$

es un **elipsoide,** para el caso en que $|y_0| < b$, la ecuación;

$$\frac{x^2}{a^2}+\frac{z^2}{c^2}=1-\frac{y_o^{\,2}}{b^2}, \tag{2.7}$$

representa una familia de elipses (o circunferencias si $a = c$) paralelas al plano xz que se forman cortando la superficie mediante planos $y = y_0$. Eligiendo, cada uno a su vez, $x = x_0$, $z = z_0$, se encontraría que los cortes de la superficie son elipses paralelas a los planos yz y xy, respectivamente. Ver figura 2.4. En la tabla 2.1 se muestran las trazas en los planos coordenados.

Plano Coordenado	Traza
$xy\ (z = 0)$	Elipse: $\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$
$xz\ (y = 0)$	Elipse: $\frac{x^2}{a^2}+\frac{z^2}{c^2}=1$
$yz\ (x = 0)$	Elipse: $\frac{y^2}{b^2}+\frac{z^2}{c^2}=1$

Tabla 2.1. Trazas en los planos Coordenados.

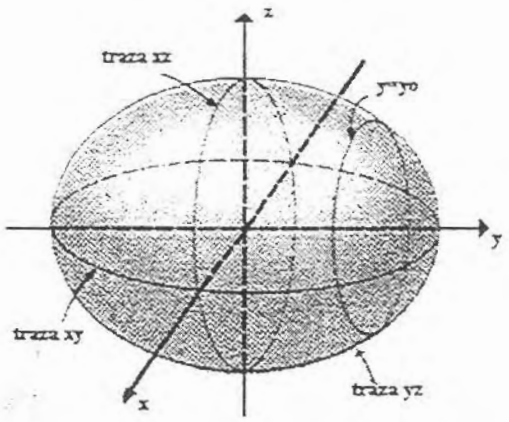


Figura 2.4. Gráfica Característica de una elipsoide.

2.3.2 Hiperboloide de una Hoja

La gráfica de una ecuación de la forma:[9]

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}-\frac{z^2}{c^2}=1\ ,\quad a,\ b,\ c>0, \tag{2.8}$$

se llama **hiperboloide de una Hoja.** En este caso, un plano $z = x_0$, paralelo al plano xy, corta la superficie en secciones transversales elípticas (o circulares, si $a = b$). Las ecuaciones de estas elipses son;

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1+\frac{z_0^{\,2}}{c^2}, \tag{2.9}$$

La tabla 2.2 presenta las trazas en los planos coordenados y la figura 2.5 representa una hiperboloide de una hoja.

Plano Coordenado	Traza
$Xy\ (z = 0)$	Elipse: $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$
$Xz\ (y = 0)$	Hipérbola: $\frac{x^2}{a^2} - \frac{z^2}{c^2} = 1$
$Yz\ (x = 0)$	Hipérbola: $\frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$

Tabla 2.2. Trazas en los planos coordenados.

La figura 2.5 representa una gráfica característica de una hiperboloide de una hoja.

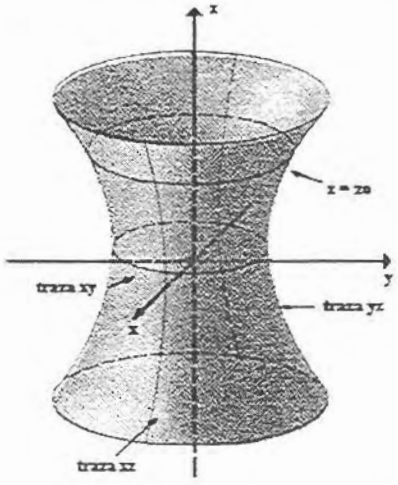


Figura 2.5. Hiperboloide de una Hoja.

2.3.3 Hiperboloide de dos Hojas

La gráfica de una ecuación de la forma:[9]

$$-\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1, \quad a, b, c > 0, \tag{2.10}$$

se llama **hiperboloide de dos hojas**, para $|y_0| > b$ la ecuación;

$$\frac{x^2}{a^2} + \frac{z^2}{c^2} = \frac{y_0^2}{b^2} - 1, \qquad (2.11)$$

describe la curva elíptica de intersección de la superficie con el plano $y = y_0$.

La tabla 2.3 presenta las trazas en los planos coordenados.

Plano Coordenado	Traza
Xy (z = 0)	Hipérbola: $-\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$
Xz (y = 0)	Ninguna.
Yz (x = 0)	Hipérbola: $\frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$

Tabla 2.3. Trazas en los planos coordenados.

La figura 2.6 representa una gráfica característica de una hiperboloide de dos hojas.

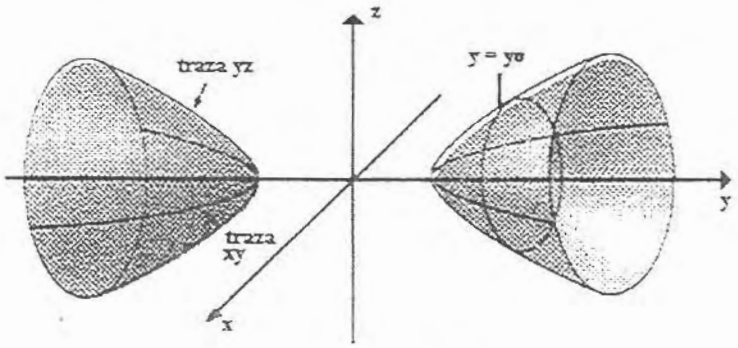


Figura 2.6. Hiperboloide de dos Hojas.

2.3.4. Paraboloide

La gráfica de una ecuación de la forma: [9]

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=cz, \tag{2.12}$$

se llama **paraboloide**. En la figura 2.7 se nota que para $c > 0$, los planos $z > 0$, paralelos al plano xy, cortan la superficie en elipses cuyas ecuaciones son;

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=cz_0, \tag{2.13}$$

La Tabla 2.4 presenta las trazas en los planos coordenados.

Plano Coordenado	Traza
Xy (z = 0)	Punto
Xz (y = 0)	Parábola: $\frac{x^2}{a^2}=cz$
Yz (x = 0)	Parábola: $\frac{y^2}{b^2}=cz$

Tabla 2.4. Trazas en los planos coordenados.

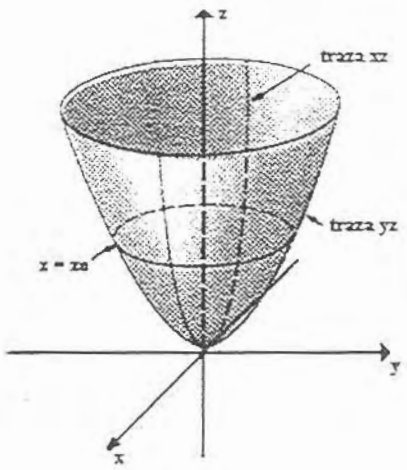


Figura 2.7.Paraboloide.

2.3.5 Paraboloide Hiperbólico

La gráfica de una ecuación de la forma:[9]

$$\frac{y^2}{a^2} - \frac{x^2}{b^2} = cz, \qquad a, b > 0 \tag{2.14}$$

se conoce como **paraboloide hiperbólico**, para $c > 0$, los planos $z = z_0$, paralelos al plano xy, cortan la superficie en hipérbolas cuyas ecuaciones son;

$$\frac{y^2}{a^2} - \frac{x^2}{b^2} = cz_0, \tag{2.15}$$

La figura 2.8 muestra la forma característica de silla de montar de un paraboloide hiperbólico.

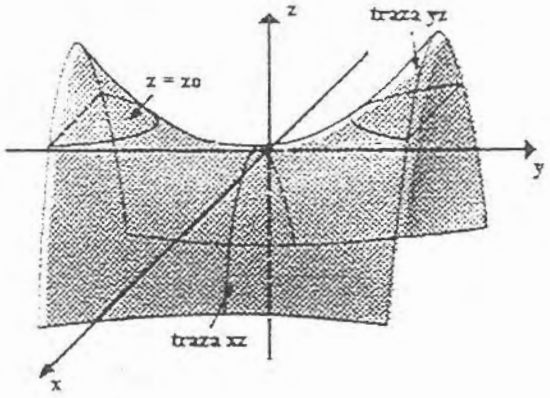


Figura 2.8. Paraboloide Hiperbólico.

La Tabla 2.5 presenta las trazas en los planos coordenados.

Plano Coordenado	Traza
Xy (z = 0)	Rectas: $\pm\frac{a}{b}x$
Xz (y = 0)	Parábola: $-\frac{x^2}{b^2}=cz$
Yz (x = 0)	Parábola: $\frac{y^2}{a^2}=cz$

Tabla 2.5. Trazas en los planos coordenados.

2.3.6 Toro o Toroide.

Es la superficie engendrada por una circunferencia que gira alrededor de una recta de su plano que no la corta.[7] Ver figura 2.9.

Considerando el eje *oz* como eje de giro y la circunferencia situada en el plano *yz* como centro sobre el eje *oy*.

$$\left.\begin{array}{c}(y-a)^2+z^2=R^2\\ x=0\end{array}\right\}\ C. \qquad (2.16)$$

en donde $a > R$

Un punto $(0, y, z)$ de C se trasforma, al girar, en otro (X, Y, Z) tal que;

$$\left.\begin{aligned} y &= X^2 + Y^2 \\ z &= Z \end{aligned}\right\} \tag{2.17}$$

Por lo tanto la ecuación de la superficie queda defina por; [7]

$$\left(\sqrt{X^2 + Y^2} - a\right)^2 + Z^2 = R^2, \tag{2.18}$$

o bien;

$$\left(\sqrt{x^2 + y^2} - a\right)^2 + z^2 = R^2, \tag{2.19}$$

desarrollando el binomio cuadrado;

$$x^2 + y^2 + z^2 - 2a\sqrt{x^2 + y^2} + a^2 = R^2, \tag{2.20}$$

se obtiene;

$$\left(x^2 + y^2 + z^2 + a^2 - R^2\right)^2 = 4a^2\left(x^2 + y^2\right) \tag{2.21}$$

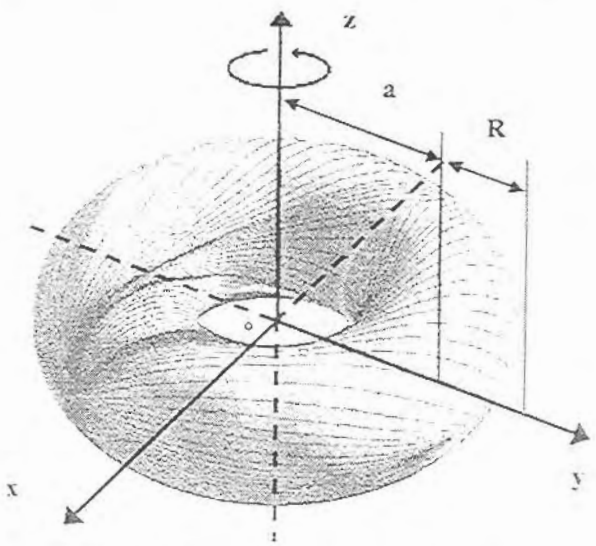


Figura 2.9. Representación de un toroide.

2.4 Superficies de Revolución.

Una superficie de revolución es la engendrada por una línea (generatriz) que gira alrededor de una recta fija (*Llamada eje de Revolución*).[9]

Los puntos de la generatriz describen circunferencias cuyos centros están en el eje y cuyos planos son normales al eje, se llaman **paralelos**.

Los planos que pasan por el eje determinan en la superficie secciones que son líneas iguales entre sí y se llaman **meridianas**.[7]

Puede considerarse, también, como generatriz de este tipo de superficies a una circunferencia (paralelo) que se mueve recorriendo su centro sobre el eje, manteniéndose su plano siempre normal al eje y apoyándose en la generatriz.

Para deducir la ecuación general de una superficie de revolución, considérese la siguiente ecuación;

$$\frac{x-x_0}{a}=\frac{y-y_0}{b}=\frac{z-z_0}{c}, \qquad (2.22)$$

y la ecuación de la generatriz;

$$\left.\begin{aligned} F(x, y, z)&=0\\ G(x, y, z)&=0\end{aligned}\right\} \qquad (2.23)$$

La generatriz vendrá dada por la intersección de una esfera de centro (x_0, y_0, z_0) y radio variable, con un plano normal al eje, es decir;

$$\left.\begin{aligned} (x-x_0)^2+(y-y_0)^2+(z-z_0)^2&=\lambda_1\\ ax+by+cz&=\lambda_2\end{aligned}\right\} \qquad (2.24)$$

expresando que esta circunferencia corte a;

$$\left.\begin{aligned} F(x, y, z)&=0\\ G(x, y, z)&=0\end{aligned}\right\} \qquad (2.25)$$

se obtiene una relación entre los parámetros;

$$\omega(\lambda_1, \lambda_2) = 0. \tag{2.26}$$

Eliminando λ_1 y λ_2 entre esta ecuación y las ecuaciones del paralelo se tiene;

$$\varphi\left[(x-x_0)^2+(y-y_0)^2+(z-z_0)^2, ax+by+cz\right]=0. \tag{2.27}$$

2.5 Superficie Óptica.

Una **superficie óptica** es la frontera entre dos medios de diferente índice de refracción, definida como una superficie de revolución alrededor del eje óptico, mediante la siguiente expresión;[8]

$$Z = \frac{CS^2}{1+\left(1-(K+1)C^2S^2\right)^{1/2}}, \tag{2.28}$$

en donde;

$$S^2 = X^2 + Y^2, \tag{2.29}$$

$$C = 1/r, \tag{2.30}$$

donde r es el radio de curvatura de la región paraxial (cerca del origen). La figura 2.10, muestra un representación de una superficie óptica.

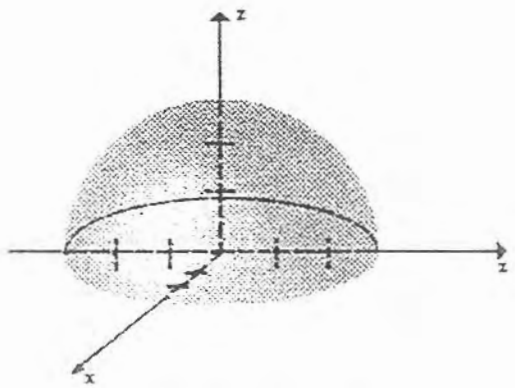


Figura 2.10. Superficie Optica.

2.6 Superficies Cónicas.

Una **superficie cónica** es una superficie de revolución [8], y dependiendo de su constante de conicidad (K), se tienen las siguientes formas, según la tabla 2.1.

Hiperboloide	$K < -1$
Paraboloide	$K = -1$
Esferoide alargado	$-1 < K < 0$
Esfera	$K = 0$
Esferoide achatado	$K > 0$

Tabla 2.6. Valores de la constante de conicidad.

Se puede observar que no existe constante de conicidad para superficie de curvatura cero, es decir, para un plano. Las figuras 2.11 y 2.12 representan las figuras correspondientes a la tabla 2.6.

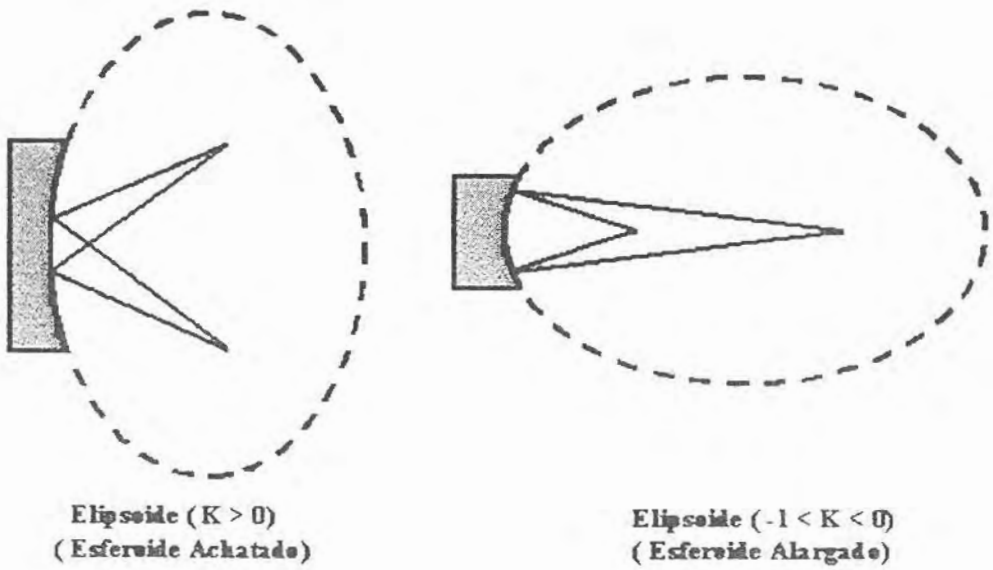


Figura 2.11. Parámetros de las Cónicas

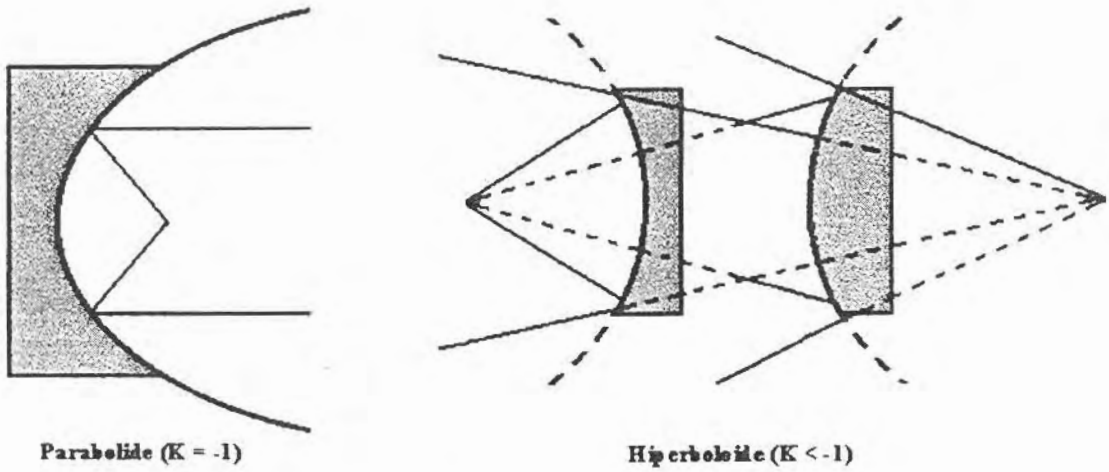


Figura 2.12. Parámetros de las Cónicas.

2.6.1 Derivación de la fórmula para las Cónicas.

Con el fin de simplificar la formulación, los planteamientos se llevarán a cabo para un plano solamente, sin que este hecho implique perder generalidad, ya que es muy sencillo pasar del caso bidimensional que se presenta, al caso tridimensional (en las siguientes fórmulas es necesario substituir sólo x^2 por $s^2 = x^2 + y^2$).[10]

Para obtener la ecuación para las cónicas, se deriva la fórmula para el caso particular de la circunferencia. Una vez logrado lo anterior, la formulación que se desarrolla para las cónicas, tenderá a buscar la misma estructura de la ecuación (2.31).

En coordenadas rectangulares, la ecuación de una circunferencia, cuyo centro está a lo largo del eje z, está dada por;

$$X^2 + (z - r)^2 = r^2, \tag{2.31}$$

donde r, es el radio de curvatura o distancia del centro a cualquier punto sobre la superficie.

Despejando z la ecuación (2.31) se obtiene;

$$z = \frac{Cx^2}{1 + \sqrt{1 - x^2C^2}}, \tag{2.32}$$

donde $C = 1/r$.

La representación de la ecuación (2.32) está mostrada en la figura 2.13 y de la misma se explica el signo positivo del radical del denominador de la ecuación (2.32). La conveniencia de expresar en óptica las curvas de los frentes de onda o superficies ópticas[10] como se muestra en la figura 2.13, es porque generalmente se trabaja con los casquetes o secciones de las superficies y consecuentemente con los frentes de onda salientes del sistema. Cabe mencionar que los ejes de simetría respectivos de cada superficie, idealmente se suponen alineados respecto a un eje de simetría común, denominado eje óptico.[10]

En lo subsecuente se derivará una expresión semejante a la ecuación (2.32) pero ahora para las cónicas, donde, como ya se mencionó, el caso de la circunferencia deberá estar contenido como un caso particular. Sin embargo, los detalles de la derivación no se presentan en este caso.

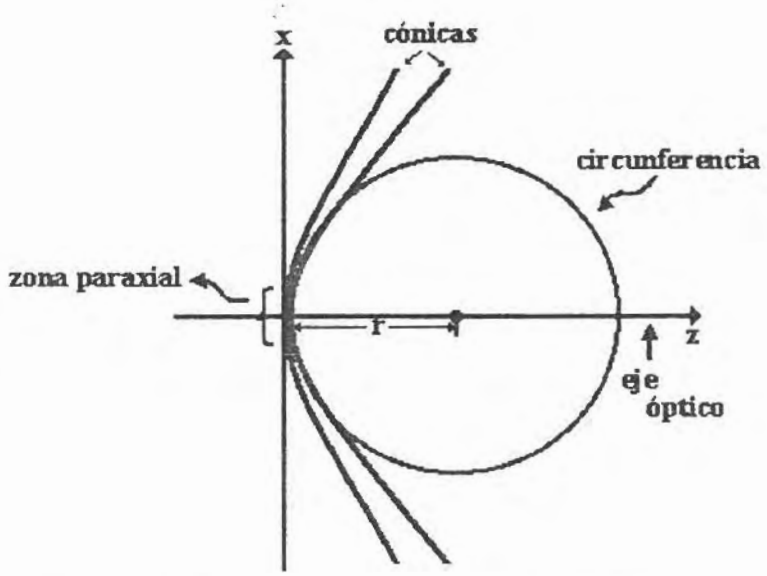


Figura 2.13.[10] Diferencias entre las secciones cónicas y la circunferencia, tomado como referencia esta última.

A partir de la ecuación general de segundo grado;

$$a_{11}x^2 + a_{12}xz + a_{22}z^2 + 2a_{14}x + 2a_{24}z + a_{32} = 0, \qquad (2.33)$$

se desea obtener una ecuación muy semejante, en primer lugar, a la ecuación (2.31) para de allí pasar a otra parecida a la ecuación (2.32). Para lograr este objetivo es necesario eliminar los términos cruzados xy, y el correspondiente a y. Esto se puede hacer mediante

una rotación y traslación adecuadas de los ejes [10], de tal manera que se obtiene una expresión simplificada dada por;

$$x^2 + \lambda z^2 + 2\mu z + F = 0. \tag{2.34}$$

Los valores de los coeficientes λ, μ y F pueden ser expresados en función de algunos de los parámetros conocidos de las cónicas, de tal manera que en referencia a la figura 2.13, es posible obtener las siguientes expresiones;

$$\left.\begin{aligned} \lambda &= \frac{b^2}{a^2}, \\ \mu &= -\frac{b^2}{a^2}, \\ F &= 0, \end{aligned}\right\} \tag{2.35}$$

Sin embargo, en lugar de sustituir los resultados de las ecuaciones (2.35) en la ecuación (2.34), se expresa λ y μ en función de otros parámetros generalmente empleados en óptica. Estos parámetros son la constante de conicidad $K = -e^2$, donde e = *excentricidad*, y el radio de curvatura r de la superficie en su zona paraxial (región cercana al eje óptico) [10]. Así se puede escribir;

$$\left.\begin{aligned} \lambda &= 1 + K, \\ \frac{b^2}{a} &= \frac{1}{c}, \end{aligned}\right\} \tag{2.36}$$

El signo de b^2/a de la ecuación (2.36) se obtiene, considerando la curvatura c de la superficie con signo positivo, cuando el centro de curvatura está a la derecha del vértice. Substituyendo los valores de λ y μ obtenidos en las ecuaciones (2.35) y (2.36), dentro de la ecuación (2.34) y al mismo tiempo resolviendo la ecuación de segundo grado obtenida para z se obtiene;

$$z = \frac{1 - \left[1 - (1+K)c^2x^2\right]^{1/2}}{(1+K)c} \tag{2.37}$$

Donde el significado del signo negativo del paréntesis rectangular en el numerador se debe a que sólo se usa la rama negativa de la solución de la cuadrática, que al mismo tiempo mantiene la convención de signos explicada anteriormente.

Para evitar el caso singular que se presentaría en la ecuación (2.37) cuando $c = 0$ (superficies planas); la misma ecuación (2.37) se trasforma en;

$$z = \frac{cs^2}{1+\left[1-(K+1)c^2s^2\right]^{1/2}}, \tag{2.38}$$

que se obtiene a partir de dicha ecuación al multiplicar numerador y denominador por $\left\{1+\left[1-(1+K)c^2s^2\right]^{1/2}\right\}$. Por lo tanto las cónicas quedan representadas[10] por la ecuación (2.38).

2.6.2 Determinación de los radios de curvatura para las diferentes zonas de una superficie cónica.

En el caso de una circunferencia, es fácil de visualizar que el radio de curvatura r tiene el mismo valor para cualquier zona de un casquete de la superficie, y que la localización de las coordenadas del centro de curvatura también es un punto único.

Luego entonces, en el caso de las cónicas; ¿Se tiene también un solo radio de curvatura?.[10]

Para encontrar respuesta a la pregunta planteada, es necesario recurrir a algunos resultados de la geometría analítica. Así se tiene que la curvatura c de una superficie cualquiera esta definida como la razón de cambio de la inclinación de la tangente al recorrer un determinado arco de la curva[18], ver figura 2.14.

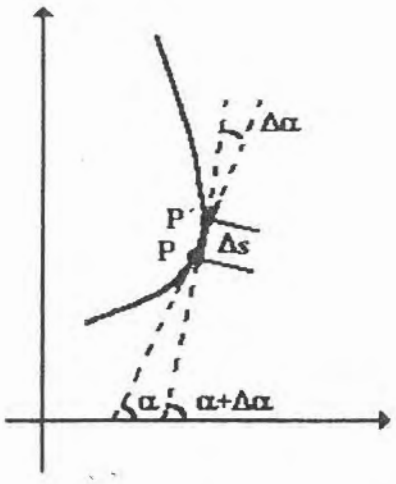


Figura 2.14.[10] Definición de la curvatura en un punto cualquiera de una superficie en función de la razón de cambio de la tangente.

Es decir;

$$c = \underbrace{lim}_{\Delta \to 0} \frac{\Delta \alpha}{\Delta s} = \frac{d\alpha}{ds} \tag{2.39}$$

Es a partir de esta definición, y para el caso de una superficie cualquiera, que se tiene la ecuación (2.40), en donde la curvatura para cierto punto o zona está en función de la primera y segunda derivadas z' y z''; [10]

$$c = \frac{z''}{\left(1 + z'^2\right)^{3/2}}, \tag{2.40}$$

si se aplica lo anterior a la ecuación (2.38) se obtiene como resultados;

$$\left.\begin{aligned} z' &= \frac{cx}{\left[1 - (K+1)c^2x^2\right]^{1/2}}, \\ z'' &= c\left[1 - (K+1)c^2x^2\right]^{-3/2} \end{aligned}\right\} \tag{2.41}$$

Por lo tanto, si se considera que el radio de curvatura R es el inverso de la curvatura c ($c = 1/R$), se tiene que el caso general de R, para cualquier zona de una cónica, puede ser expresado como;

$$R = \frac{1}{c}\left[1 - Kc^2x^2\right]^{3/2}, \qquad (2.42)$$

donde R es el radio tangencial[8] de la superficie.

La figura 2.15 muestra la gráfica de cómo cambia el radio de una superficie cónica. Analizando la ecuación (2.42) es fácil observar la dependencia de R, de la zona correspondiente para una valor de x que se esta considerando. Por otra parte, para el caso particular de una circunferencia, cuando $K = 0$, se tiene que $R = 1/c = r$. Es decir, para esta última situación se tiene un radio único e independiente de la zona que se considere. Se hace notar también que para los casos de secciones en que se encuentran cerca del eje óptico, el producto (cx^2) tiene valores muy pequeños y, por lo tanto, el radio de curvatura es casi el mismo para cualquier cónica, y dicho radio de curvatura es denominado generalmente en óptica como radio de curvatura paraxial. Con referencia a la figura 2.13, se tiene que cerca del eje óptico todas las cónicas tienden a la circunferencia.

Por lo tanto, como respuesta a las preguntas antes planteadas, se tiene que no existe un solo valor para el radio de curvatura de una cónica.[10]

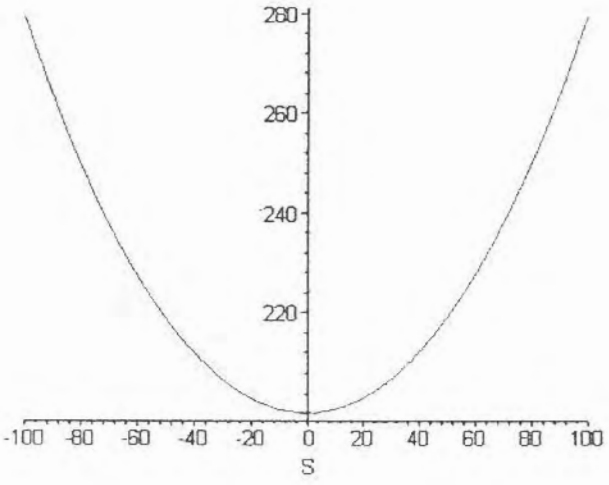


Figura 2.15. Gráfica que representa el cambio del radio de Curvatura de una superficie cónica, K=-1, C=0.005. (Parabolide).

2.7 Superficies Asféricas.

Las superficies asféricas son superficies ópticas que no son ni esféricas ni planas, una esférica tiene un solo parámetro de forma, el cual es el radio de curvatura ***R***, una asférica, por el contrario, tiene varios parámetros de forma.[6]

Las superficies asféricas ofrecen una gran variedad de posibilidades de imágenes, por ello fueron estudiadas por muchos científicos como Kepler en 1611, Descartes en 1638 y Huygens en 1678. Por ejemplo haciendo uso de la ley de la refracción, Descartes calculó con gran precisión la forma de lentes asféricas sobre un punto axial, lo cual no es posible de realizar con superficies esféricas. Además diseñó una maquina para pulir lentes asféricas.

La primer superficie asférica usada en la practica, aparece a mediados del siglo XVIII, y fue usada en telescopios.[6]

La necesidad de contar con sistemas de imágenes con mejor calidad (definición), ha dado lugar al surgimiento de formas más complicadas en los lentes y de varios tipos. Las últimas décadas han marcado el desarrollo de las superficies asféricas, principalmente por el progreso de la computación, por nuevas técnicas de manufactura, y nuevos métodos de prueba han tenido que desarrollarse para las mismas.

De esta manera la fabricación de superficies asféricas con formas complicadas se vuelve más posible y efectivo. Por otro lado, la demanda de superficies asféricas esta incrementándose día a día, nuevas aplicaciones se están desarrollando debido a que muchos problemas ópticos, son más fáciles de resolver usando superficies asféricas que esféricas.

Por ejemplo, una sola superficie asférica (SA) puede remplazar a una serie de superficies esféricas, para realizar el mismo o mejor trabajo, permitiendo así, que los sistemas ópticos sean más compactos y livianos.

Las superficies de revolución son usadas como superficies reflejantes o refractantes.[6] Los tipos más simples son superficies cuádricas. Como se mencionó en la sección 2.5, una **superficie óptica** es la frontera entre dos medios de diferente índice de refracción, definida como una superficie de revolución alrededor del eje óptico, mediante la siguiente expresión;[8]

$$Z = \frac{CS^2}{1+\left(1-(K+1)C^2S^2\right)^{1/2}} + A_1S^4 + A_2S^6 + A_3S^8 + A_4S^{10}, \qquad (2.43)$$

en donde;

$$S^2 = X^2 + Y^2, \qquad (2.44)$$

$$C = 1/r, \qquad (2.45)$$

donde r es el radio de curvatura de la región paraxial (cerca del origen), A_1, A_2, A_3 y A_4 son los coeficientes de deformación de las asféricas, $K = -e^2$, e = *excentricidad*.

2.7.1 Radios de curvatura Sagital y Tangencial.

El radio de curvatura de una superficie asférica de revolución en un punto determinado viene dado por el radio de curvatura, evaluado en ese mismo punto de la curva, que es generada al intersectar la superficie con un plano.[8]

De esta definición se desprende que, en general, existe un número muy grande de radios de curvatura en un mismo punto de la superficie. Para el caso que se está tratando, solo interesan los radios de curvatura que se obtienen cuando el plano que corta a la superficie asférica es perpendicular a ella. En particular, solo se consideran los dos siguientes;[8]

1) Cuando el plano que corta la superficie contiene al eje de simetría (eje óptico) de la superficie asférica. Este plano se denomina *plano tangencial* y el radio de curvatura que se obtiene es el **tangencial**.

2) Cuando el plano que corta la superficie es perpendicular al plano tangencial. En este caso el radio de curvatura es el **sagital**.

2.8 Conclusiones.

En este capítulo, se describió la forma analítica de las superficies que comúnmente se utilizan en componentes ópticos. También se observó que la mayoría de estas superficies son superficies de revolución, pudiéndose describir a estas de manera general con las superficies asféricas y considerando a las superficies cónicas como parte de ellas (asféricas).

CAPÍTULO III

ALGORTIMOS GENÉTICOS

Introducción.

La **genética**, es una rama de la biología que se encarga del estudio de cómo se transmiten los caracteres físicos, bioquímicos y de comportamiento de padres a hijos.[11]

Este término fue acuñado en 1906 por el biólogo Británico ***William Bateson*** [11], el cual estaba convencido de que todos los organismos vivos procedían de un grupo de antecesores comunes, e intentó comprender en qué consistía el proceso de la evolución. Observó que, en general, las especies son muy diferentes entre sí, así como que determinados rasgos distintivos de un tipo de organismo pueden aparecer o desaparecer súbitamente de una generación a otra. ***Bateson*** llegó a la conclusión de que la evolución se produce en gran medida por una serie de saltos repentinos y discontinuos, y no gradual y progresivamente.

En el presente capítulo se presenta una breve reseña sobre la teoría de la evolución propuesta por ***Darwin***, con el objetivo de fundamentar los principios sobre los cuales se sustentan los **Algoritmos Genéticos**. Posteriormente se describirá sus diferencias con respecto a los métodos tradicionales de optimización, cuando es viable aplicar un algoritmo genético, los algoritmos genéticos, algunas definiciones, componentes básicos, y los esquemas o bloques construidos. Por último se presentan las conclusiones.

3.1 Historia de la evolución.

La Tierra se formó hace aproximadamente unos 4.000 a 5.000 millones de años. Existen fósiles de criaturas microscópicas del tipo de las bacterias que prueban que surgió la vida hace unos 3.000 millones de años. En algún momento entre estas dos fechas —la evidencia molecular supone que hace cerca de 4.000 millones de años— debió tener lugar el increíble suceso del origen de la vida. No se sabe qué ocurrió, aunque los teóricos coinciden en que la clave fue la aparición espontánea de seres que se autorreplicaban, es decir, algo equivalente a los *genes* en sentido general [11].

Es probable que al principio la atmósfera de la Tierra contuviera metano, amoniaco, dióxido de carbono y otros gases que abundan aún en otros planetas del sistema solar. Los químicos han reconstruido en los laboratorios estas condiciones primitivas a un nivel experimental. Si se mezclan los gases adecuados con agua en un matraz, y se añade energía mediante una descarga eléctrica (simulando la iluminación primitiva), se sintetizan de forma espontánea sustancias orgánicas. Entre éstas se cuentan, en una proporción significativa, aminoácidos (unidades que construyen las proteínas, incluyendo todas las enzimas importantes que controlan los procesos químicos de la vida), purinas y pirimidinas (unidades que forman el ARN y ADN). Parece probable que al principio de la existencia de la tierra sucediera algo similar. Por consiguiente, el mar podría haber sido un "caldo" de compuestos orgánicos pre-biológicos.[12]

Como es natural, el hecho de que las moléculas orgánicas aparecieran en este "caldo" primitivo, no es suficiente. El paso más importante fue la aparición de moléculas que se autorreplicaban, capaces de producir copias de sí mismas. Hoy, la molécula más conocida que se autorreplica es el ácido desoxirribonucleico (ADN). La creencia de que el propio ADN no podría haber estado presente en el origen de la vida está muy extendida, ya que su replicación depende demasiado de estructuras muy especializadas que no pudieron existir antes del inicio de la propia evolución. El ADN ha sido descrito como una molécula de 'alta tecnología' que apareció con toda probabilidad algún tiempo después del origen de la vida. Tal vez la molécula con la que está emparentada, el ácido ribonucleico (ARN), que aún desempeña varias funciones vitales en las células vivas, fue la molécula autorreplicativa original, o tal vez ésta fue un tipo de molécula diferente. Una vez que las moléculas autorreplicativas se habían formado por casualidad, pudo haberse iniciado algo parecido a ***la selección natural darwiniana***: las variaciones presentes en las poblaciones podrían tener su origen en errores aleatorios en el copiado. Las variantes con una replicación especialmente buena habrían predominado automáticamente en el caldo primitivo, mientras que aquellas que no se replicaron, o que lo hicieron de forma errónea, estarían en una proporción relativamente menos numerosa.

Estos mecanismos fueron construidos probablemente mediante la manipulación de otras moléculas, tal vez proteínas. Otros mecanismos manipulados fueron aquellas estructuras previas a las membranas que proporcionaron espacios circunscritos donde incluir las reacciones químicas. Pudo haber sido poco después de este estadio cuando las

criaturas simples del tipo de las bacterias dieron lugar a los primeros fósiles hace más de 3.000 millones de años.

El resto de la evolución puede ser considerada como una continuación de la selección natural de las moléculas replicativas, ahora denominadas genes, debida a su capacidad para construir por sí mismas estructuras eficaces (cuerpos celulares y multicelulares) para su propia supervivencia y reproducción. Tres mil millones de años es un periodo de tiempo largo, y parece que ha sido lo suficientemente prolongado como para haber dado origen a estructuras tan increíblemente complejas como el cuerpo de los vertebrados y de los insectos. Con frecuencia, se hace referencia a los genes como al medio que emplean los cuerpos para reproducirse. Esto es a primera vista innegable, aunque es más cierto el hecho de que los cuerpos son el medio que utilizan los genes para reproducirse.[12]

A lo largo de nuestra existencia, y conforme vamos creciendo es inevitable darnos cuenta de la gran diversidad de vida que existe a nuestro alrededor, la capacidad con la que están dotados los organismos vivos para sobrevivir y multiplicarse, un claro ejemplo es el mimetismo en ciertos insectos, los cuales imitan a otras especies en color, forma, sonidos, etc., para así poder evitar su depredación, con lo cual está asegurando la descendencia de su especie.

Como se manifiesta en la teoría de la evolución de ***Darwin***, el medio ambiente determina el grado de éxito en la reproducción de individuos y grupos de organismos. La **selección natural**, es el proceso por el cual los cambios ambientales determinan de forma variable el éxito reproductivo entre los individuos de una población de organismos con características, formas, movimientos, etc., diferentes y heredables a sus descendientes. Las características que bloquean el éxito reproductivo se hacen menos frecuentes de generación en generación, garantizando así que los individuos resultantes son los que mejor se adaptarán a su medio ambiente.

De esta manera, la selección natural tiende a mejorar la adaptación al mantener aquellas adaptaciones que resultan favorables en un entorno estable, o bien, al favorecer adaptaciones en la dirección adecuada ante cambios ambientales. Esta teoría está contenida en el histórico tratado de ***Charles Darwin*** *"El origen de las especies"* [12], publicado en 1859. En resumen, los seres vivos más aptos a un cambio sobreviven, los que no, desaparecen.

La teoría de la evolución tiene como características principales, dentro de sus fundamentos:

- La evolución es un proceso que se ejerce sobre los cromosomas.
- La selección natural hace que los cromosomas codifiquen seres más aptos.
- La reproducción es el proceso mediante el cual se realiza la evolución, garantizando la mezcla y la recombinación de genes entre los descendientes.
- La evolución no tiene memoria, es decir, todo individuo que se encuentra funcionando correctamente en su medio ambiente, es gracias al conjunto de cromosomas que portan.

3.2 El Algoritmo genético.

Un ***Algoritmo Genético*** (AG), es una técnica de búsqueda de solución a problemas basada en los principios de la selección natural y supervivencia propuestos por ***Darwin***, es decir utilizan una analogía directa del fenómeno evolutivo en la naturaleza para resolver problemas.

Un investigador de la Universidad de Michigan llamado ***John Holland*** [13], interesado y consciente de la importancia de la evolución natural, desarrolló a fines de los 60´s una técnica en la cual se imitaron todas las características de la evolución en un algoritmo para computadora, es decir con esta técnica se podrían resolver problemas de manera similar a como lo hace la naturaleza a través de la evolución. En los primeros estudios desarrollados por ***Holland***, se plantea la perspectiva de abstraer y explicar los procesos de adaptación en sistemas naturales, diseñando un sistema artificial que, mediante programación, emule los mecanismos de los sistemas naturales, demostrando con esto que, bajo un control adecuado y con ciertas transformaciones, se mejora la calidad de la población conforme evoluciona. Así ***Holland***, empezó a trabajar con algoritmos que manejaran cadenas de bits, a las cuales las llamó cromosomas.

A esta técnica inventada por ***Holland*** se le llamo originalmente "planes reproductivos" pero posteriormente se hizo popular bajo el nombre de "algoritmos genéticos". Una definición formal de un algoritmo genético, propuesta por ***John Koza***; [13]

"Es un algoritmo que trasforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su amplitud."

Anselmo Pérez Serrada, propone la siguiente definición;[14]

"Los Algoritmos Genéticos son métodos de búsqueda ciega de soluciones cuasi-óptimas. En ellos se mantiene una población que representa a un conjunto de posibles soluciones la cual es sometida a ciertas transformaciones con las que se trata de obtener nuevos candidatos y a un proceso de selección sesgado a favor de los mejores candidatos."

3.2.1 Generalidades del Algoritmo Genético.

Existen dos mecanismos que ligan el problema a resolver con el algoritmo genético: el primero es la codificación del problema en un cromosoma y el otro es la función de evaluación que se le aplicará a cada cromosoma dependiendo del problema.[3]

Si un problema puede ser representado por un conjunto de parámetros (conocidos como genes), estos pueden ser unidos para formar una cadena de valores (cromosoma), a este proceso se le llama codificación. En genética este conjunto representado por un cromosoma en particular es referido como genotipo, este contiene la información necesaria para construir un organismo, conocido como fenotipo. Estos mismos términos se aplican en Algoritmos Genéticos, por ejemplo, si se desea diseñar un puente, el conjunto de parámetros especificando el diseño es el genotipo, y la construcción final es el fenotipo. La adaptación de cada individuo depende de su fenotipo, el cual se puede inferir de su genotipo, es decir, puede calcularse desde el cromosoma utilizando la función de evaluación.[15]

Por ejemplo, si se tiene un problema de maximizar una función de tres variables, $f(x,y,z)$, se podría representar cada variable por un número binario de 10 bits, obteniéndose un cromosoma de 30 bits de longitud y 3 genes.

La técnica más común para la codificación del problema es a través de cadenas binarias, cabe mencionar que esta no es la única manera de hacerlo, se pueden utilizar también números reales, letras, etc. La utilización de cadenas binarias es lo que ha tenido más auge debido a que es más sencillo de implementar además de que es la propuesta original de ***Holland***.

La función de evaluación, es la conexión entre el algoritmo genético y el problema que se va a resolver, la función de evaluación tiene como entrada a un cromosoma y como salida uno o varios números, los cuales son una medida del funcionamiento del cromosoma en el contexto del problema ha ser resuelto [3] (*Figura 3.1*)

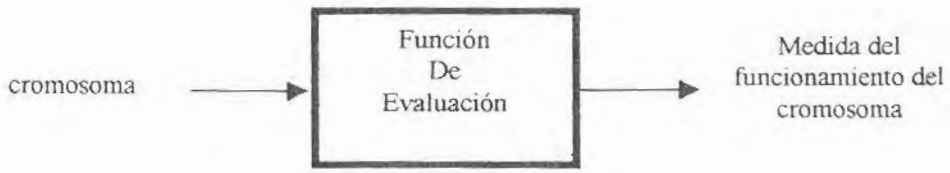


Figura 3.1. Función de Evaluación

Haciendo una analogía, la función de evaluación del algoritmo genético, es lo que correspondería al medio ambiente en la evolución natural. La iteración de un cromosoma con la función de evaluación proporciona una medida de aptitud que el algoritmo genético usa cuando se lleva a cabo la etapa de reproducción.

Una población inicial de cromosomas, mejorará cuando los padres sean remplazados por mejores y mejores hijos, es decir conforme avance el proceso evolutivo, el mejor individuo de la población final será una solución altamente confiable para el problema. A continuación se muestra la descripción de un algoritmo genético [16] (*Figura 3.3*).

En el primer paso se genera una población inicial de cromosomas aleatoriamente, la cual consiste en una serie de cadenas binarias, una por cada miembro de la población, (*Figura 3.2*).

011011010 111001101 100001101 101111110 1010101011

Figura 3.2. Población de 5 individuos.

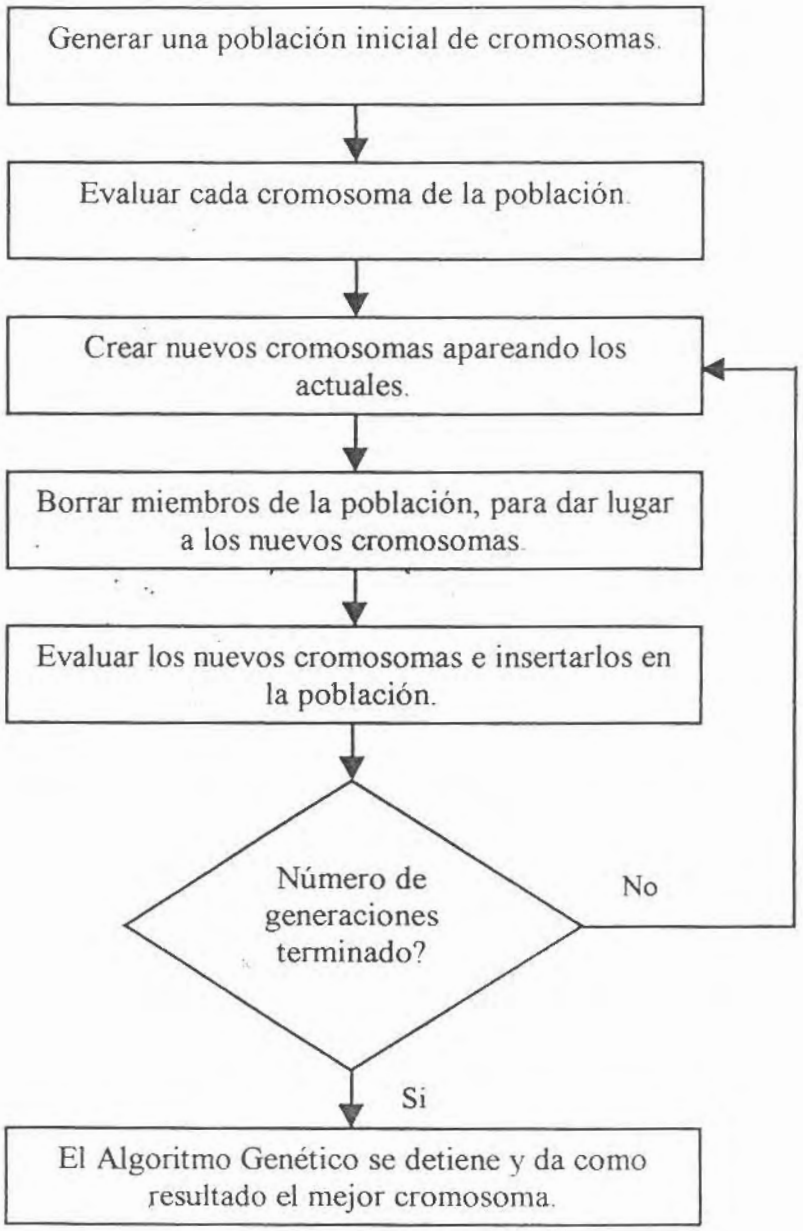


Figura 3.3. Descripción General de un Algoritmo Genético

En la figura 3.2 se presenta una población de cinco cromosomas, los cuales tiene una longitud de nueve bits cada uno.

La población es generada aleatoriamente para permitir la diversidad de la búsqueda y con el fin de contar con individuos distintos entre ellos dentro del espacio de

soluciones del problema, con la capacidad de dar origen a otra población mediante la "cruza" de dos de estos.

En la naturaleza, supongamos que el primer individuo de la figura 3.2 es una coneja, y los otros cuatro son conejos, estos cuatro conejos tiene la misma oportunidad de aparearse con la coneja, el éxito para cada conejo está basado en la "*Ley del más fuerte*", es decir el conejo más fuerte será el que consiga aparearse con la coneja.

De manera análoga un individuo de la población inicial tiene la misma oportunidad de aparearse con otro individuo, a diferencia de la "*Ley del más fuerte*" que rige a la naturaleza, el éxito de un apareamiento entre individuos dependerá del valor devuelto por la función de evaluación que se le aplica a cada individuo, es decir los individuos mejor evaluados serán los que tengan mayor probabilidad de aparearse para dar lugar a una nueva población. La cruza entre dos individuos se explicará más adelante.

En el paso dos, cada cromosoma de la población inicial, es evaluado por la función de evaluación, la cual devolverá uno o varios números, dependiendo del contexto del problema y de la misma función de evaluación, (*Figua 3.4*).

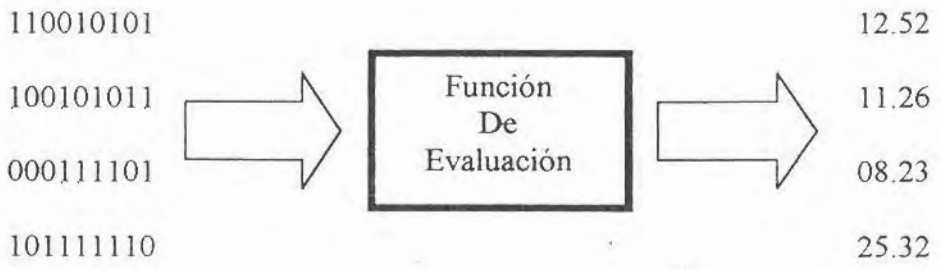


Figura 3.4. Valores devueltos por la función de evaluación.

En la figura 3.4 el primer cromosoma de la población inicial (110010101), es evaluado por la función de evaluación y esta regresa el valor de 12.52, esto sucede para cada cromosoma de la población.

En el tercer paso, se generan nuevos cromosomas a partir de los iniciales, en esta parte es donde se seleccionan los cromosomas mejores evaluados y se aparean para formar una nueva población, ejemplo; se tiene una población de diez cromosomas, de los cuales los cinco mejores evaluados, serán los que den lugar a una nueva población de diez nuevos cromosomas. La cruza de dos cromosomas dará como resultado dos nuevos cromosomas.

El paso cuatro es el que se encarga de borrar los cromosomas "viejos", con el fin de dar lugar a los nuevos cromosomas, posteriormente, en el paso cinco se evalúan los nuevos cromosomas, finalmente, si ya se cumplió con el número de generaciones, el algoritmo termina y el cromosoma mejor evaluado de la población vigente será una solución seguramente muy cercana al óptimo del problema, en caso contrario se repite el procedimiento a partir del paso tres.

3.2.2 ¿Cuándo de debe aplicar un Algoritmo Genético?

El poder de los Algoritmos Genéticos proviene del hecho de que la técnica es robusta, y puede manejar exitosamente un amplio rango de problemas, incluso algunos que son difíciles de resolver por otros métodos. Los algoritmos genéticos no garantizan que encontrarán la solución óptima al problema, pero son generalmente buenos encontrando soluciones muy cercanas a este, las cuales son aceptables a problemas en corto tiempo. Donde existan técnicas especializadas para la resolución de problemas, estas superarán fácilmente a los algoritmos genéticos tanto en velocidad como en precisión. El campo principal de aplicación es donde no existan este tipo de técnicas. Por lo que se recomienda tener en cuenta las siguientes características del problema al cuál se le quiere dar solución.[13]

- Su espacio de búsqueda, es decir sus posibles soluciones, para la solución del problema debe estar delimitado dentro de un cierto rango. Lo más recomendable es intentar resolver problemas que tengan espacios de búsqueda discretos, aunque estos sean muy grandes. Sin embargo, también es posible utilizar los AG's cuando se tengan espacios de búsqueda continuos, pero preferentemente cuando exista un rango de soluciones relativamente pequeño.
- Debe poderse definir una función de evaluación, la cual indicará que tan buena o que tan mala es una respuesta.
- Las soluciones deben codificarse de una forma que resulte relativamente fácil de implementar en una computadora.

3.2.3 Diferencias entre los Algoritmos Genéticos y los métodos tradicionales de optimización.

Los Algoritmos Genéticos por lo general manejan variables de decisión o de control principalmente, representadas como cadenas con el fin de explotar similitudes entre estas. Los métodos tradicionales, por lo regular, trabajan con sus funciones y sus variables de control directamente.[17]

Los AG's trabajan sobre una población, los métodos tradicionales regularmente trabajan con un punto específico. De tal manera que los AG's al mantener una población de puntos bien adaptados, la probabilidad de alcanzar óptimos falsos se reduce considerablemente.

Los AG's obtienen gran parte de su amplitud ignorando la información que no sea la del objetivo a optimizar. Otros métodos se basan estrictamente en tal información, y en aquellos problemas donde la información sea difícil de conseguir o no este disponible, estos métodos por lo regular fallan. Los AG's son globales porque explotan la información disponible en cualquier problema de búsqueda. Los AG's procesan las similitudes en el código subyacente junto con la información proveniente de la ordenación de las estructuras de acuerdo a sus capacidades de supervivencia en el entorno actual. Al explotar la información tan fácilmente, los AG's se aplican prácticamente a cualquier problema.

Las reglas de transición de los AG's son probabilísticas, los métodos tradicionales, cuentan normalmente con reglas de transición determinísticas. Los AG's usan el azar para guiar una búsqueda fuertemente explorada, este hecho puede parecer inusual, pero hay gran cantidad de precedentes en la naturaleza.

Los algoritmos genéticos, en resumen, difieren principalmente en cuatro aspectos con relación a los métodos tradicionales:

- Trabajan con un código de los parámetros, no con los parámetros mismos.
- Buscan simultáneamente la solución en una población de puntos, no en un solo punto.
- Utilizan únicamente la información de la función de evaluación, y no sus derivadas u otro conocimiento auxiliar.
- Utilizan reglas de transición probabilísticas, en lugar de reglas determinísticas.

Dentro de las características más importantes de los AG's se tienen; [17]

Son métodos de búsqueda:

- **Ciega**, es decir, no cuentan con ningún conocimiento particular o general del problema, de tal manera que la búsqueda se basa únicamente en los valores de la función de evaluación.
- **Múltiple**, se refiere al hecho de que buscan simultáneamente al mejor candidato entre un conjunto de estos.
- **Codificada**, debido a que no operan directamente sobre el dominio mismo del problema, sino sobre representaciones de sus elementos.

La característica esencial de los AG's según Goldberg[18], es la capacidad que tiene estos, para el intercambio estructurado de información en paralelo, en otras palabras: los AG's procesan externamente cadenas de códigos, sin embargo, lo que se procesa internamente, son las similitudes entre cadenas y de tal manera que al procesar cada una de las cadenas de la población, se están procesando a la vez todos los patrones de similitud que contienen, que son muchos más. Es por esta propiedad que los AG's son mucho más eficaces que otros métodos de búsqueda, y por lo tanto más robustos con independencia de consideraciones.

3.3 Estructura y Componentes Básicos de un Algoritmo Genético.

Cuando se ejecuta un AG's, una población de individuos, que representa a un conjunto de candidatos a posibles soluciones de un problema, es sometida a una serie de transformaciones con las que se actualiza la búsqueda y después a un proceso de selección el cual favorecerá a los mejores individuos, a cada ciclo de búsqueda – selección, se le conoce como *generación.* [16]

Al cabo de un número razonable de generaciones, se espera que el AG, arroje una solución muy próxima a la deseada, esta solución estará representada por el mejor individuo de la última generación.

Los componentes principales de un AG son: Generar la población inicial, módulo de evaluación, de selección, de cruza y el de mutación. A continuación se revisará cada uno de ellos por separado.

3.3.1 Generar Población Inicial.

El primer paso para construir un AG, es generar aleatoriamente la población inicial, la cual estará constituida por un conjunto de cromosomas, o cadenas de caracteres que representan las posibles soluciones del problema. Cabe mencionar que en la practica, por lo regular se hace uso de cadenas binarias para representar a los cromosomas, por lo que estas suelen implementarse como cadenas de bits.

3.3.2 Módulo de Evaluación.

Para cada uno de los cromosomas de esta población inicial, se le aplica la función de evaluación, con el objetivo de conocer la aptitud del cromosoma, es decir, que tan buena o mala es la solución que esta codificando. La función de evaluación depende de cada problema en particular.

3.3.3 Selección de Cromosomas.

Una vez que se conoce la aptitud de cada cromosoma, se procede al proceso de selección, en este proceso es donde se seleccionan a los cromosomas que van a reproducir y posteriormente a cruzar para dar lugar a una nueva generación. En la selección, tentativamente, los cromosomas mejor evaluados tienen mayor probabilidad de ser escogidos.

La finalidad de seleccionar a los cromosomas, es para darles más oportunidades de reproducirse a los miembros de la población que son más aptos o que tiene mejores evaluaciones, cabe mencionar que todos tienen probabilidad de ser escogidos.

Existen varios métodos para llevar a cabo el proceso de selección, a continuación se describirán los dos más usados en la práctica:[13]

• **La Rueda de la Ruleta**: Este método es muy simple, consiste en crear una ruleta en que cada cromosoma tiene asignada una fracción de esta proporcional a la medida de su aptitud.[16]

La figura 3.5 muestra un ejemplo de una población de 8 cromosomas con su respectiva evaluación (los valores de la evaluación no se refieren a una función de evaluación en particular, son arbitrarios, con la finalidad de dar sustento al ejemplo).

La primera columna contiene el número del cromosoma, la segunda la cadena binaria del cromosoma, la tercera contiene la evaluación de cada cromosoma, finalmente la cuarta la suma acumulativa de las evaluaciones.

El resultado de la selección por medio de este método, es la selección de una cadena aleatoriamente. Aunque este procedimiento es aleatorio, la oportunidad que tiene cada cadena de ser seleccionada es directamente proporcional a su evaluación. Al transcurrir las generaciones, este mecanismo elimina los individuos con menores evaluaciones y tiende a expandir el material genético de los miembros mejor evaluados. Es posible que el peor o de los peores miembros de la población sean seleccionados cada vez que se use el procedimiento, sin embargo la probabilidad de que este hecho suceda en una población es pequeña.

Cromosoma No.	Cadena Binara	Evaluación	Suma Acumulativa
1	101101101	365	365
2	110100010	418	783
3	111100101	485	1268
4	001011001	89	1357
5	101010111	343	1700
6	111110001	497	2197
7	010110001	177	2374
8	000011001	25	2399

Figura 3.5. Población de cromosomas

La suma acumulativa de las evaluaciones del ejemplo 3.5, es de 2399, en la figura 3.6 se muestran cinco números entre 0 y 2399, los cuales han sido creados aleatoriamente, así mismo se muestra el número de cromosoma que ha sido seleccionado por la técnica de la ruleta, para cada caso, el cromosoma seleccionado es el primero para el cual la suma acumulativa de las evaluaciones sea mayor o igual al número aleatorio.

Número Aleatorio	Cromo. Seleccionado
200	1
2380	8
800	3
1690	5
2200	7

Figura 3.6. Cromosoma seleccionado por la técnica de la ruleta.

Este procedimiento es llamado la rueda de la ruleta porque es equivalente a asignarle una "rebanada" de un círculo a cada miembro de la población, de tal manera que el tamaño de la rebanada sea proporcional a su evaluación, después de esto, el círculo se gira y se lanza un dardo hacia él, la cadena seleccionada es aquella donde cayo el

dardo, es claro que aquel cromosoma que tenga una "rebanada" más grande del círculo, tendrá mayor probabilidad de ser acertado por el dardo. La figura 3.7 muestra la rebanada del círculo que le corresponde a cada individuo de la figura 3.5.

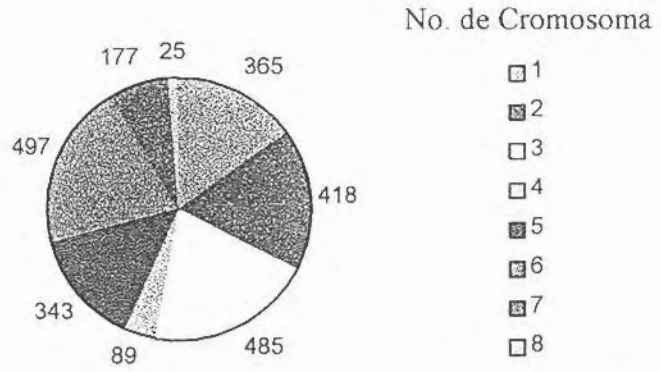


Figura 3.7. "Rebanada" del pastel correspondiente a cada cromosoma.

• **El torneo**: La idea fundamental de este método es muy sencilla, se baraja la población y después se hace competir a los cromosomas que la integran en grupos de tamaño previamente definido (por lo regular en parejas) en un torneo del que resultarán ganadores aquéllos que tengan valores de aptitud más altos. Si se efectúa un torneo binario, es decir, una competencia por parejas, entonces la población se debe barajar dos veces. Nótese que esta técnica garantiza la obtención de múltiples copias del mejor individuo entre los progenitores de la siguiente generación, en un torneo binario, los mejores individuos serán seleccionados, esta técnica es importante en las últimas etapas de optimización, ya que sirve para realizar un "ajuste fino".[19]

3.3.4 Cruza de Cromosomas.

En la naturaleza, una población de individuos lucha diariamente entre sí por la búsqueda de comida, agua, refugio, pareja, etc. Aquellos individuos que tienen más éxito en sobrevivir y en atraer pareja tienen mayor probabilidad de generar un gran número de descendientes. Por otro lado, individuos poco dotados producirán un menor número de descendientes.

Esto significa que los genes de los individuos mejor adaptados se propagarán en sucesivas generaciones hacia un número de individuos creciente. La combinación de buenas características provenientes de diferentes progenitores, seguramente tendrá como resultado descendientes "mejores", cuya adaptación es mucho mayor que la de cualquiera de sus progenitores. De esta manera, las especies evolucionan logrando unas características cada vez mejor adaptadas al entorno en el que viven.

Como se puede observar, la parte fundamental de la evolución en la naturaleza es el intercambio genético entre los individuos de una población, esto se lleva a cabo mediante la cruza o apareamiento de dos individuo.

De manera análoga, un AG recombina el material genético de dos cromosomas para crear dos hijos, los cuales formarán parte de la población de la siguiente generación.

Holland experimentó con un operador de cruzas al que denominó *cruza en un punto* [20]. Cuando se usa un solo punto de cruza, éste se escoge de forma aleatoria sobre la longitud de la cadena que representa el cromosoma, (i.e., se tiene el cromosoma 010111110, cuya longitud es de 9 bits, entonces se selecciona un número aleatorio entre 1 y 9) a partir de él se realiza el intercambio del material cromosómico de los dos individuos. En la figura 3.7 se presenta un ejemplo de una cruza;

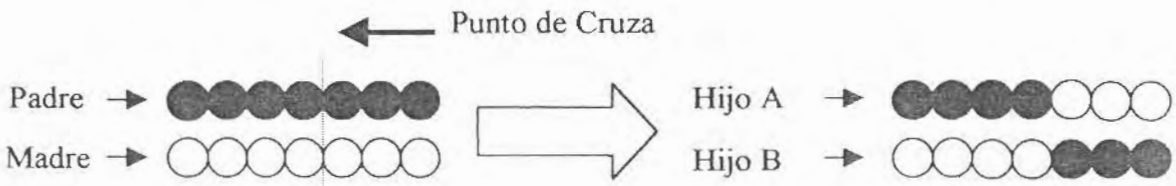


Figura 3.7. Cruza en un punto.

Una característica importante que presenta la figura 3.7, es que el operador de cruza, puede producir hijos completamente diferentes a los padres. Otra característica importante que se presenta en la cruza en un punto es que no introduce diferencias en un bit en una posición donde los padres tengan el mismo valor, en la figura 3.8, nótese que tanto los padres como los hijos tiene el mismo valor en las posiciones 3,5 y 7.

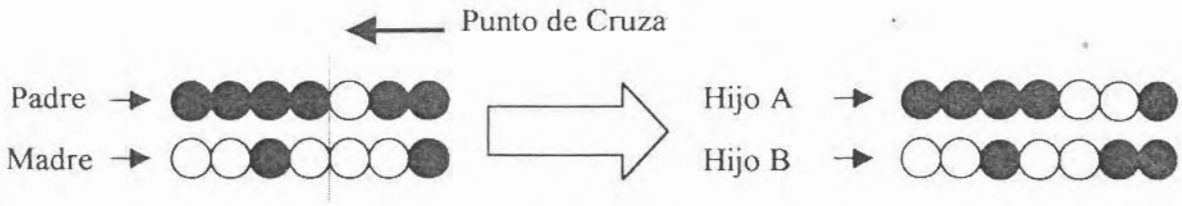


Figura 3.8. Mismos valores en posiciones 3, 5 y 7.

Existe otra forma de cruza, en la cuál se utilizan dos puntos de cruza [20], en este caso se procede de manera similar que el anterior, sin embrago el intercambio se realiza como se presenta en la figura 3.9.

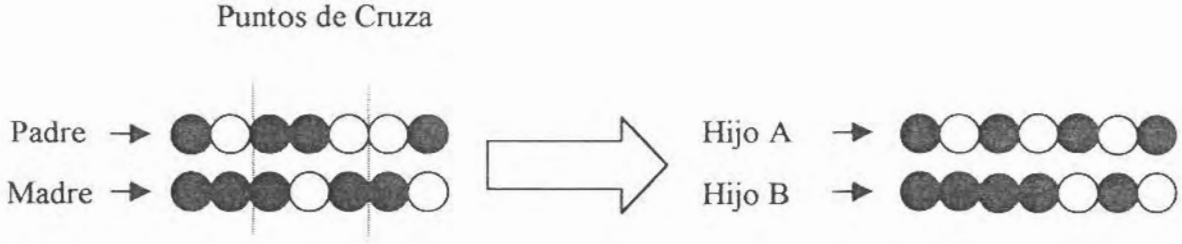


Figura 3.9. Cruza en dos puntos.

La cruza es un componente muy importante en los AG's, sin este operador, los AG's simplemente no funcionan.

3.3.5 La Mutación

Una mutación consiste en seleccionar aleatoriamente a un cromosoma de la población, posteriormente, se selecciona un gene aleatoriamente del cromosoma y se cambia su valor, es decir, si el gene es 1 se cambia a 0 y viceversa, en la figura 3.10, se observa una mutación de un cromosoma.

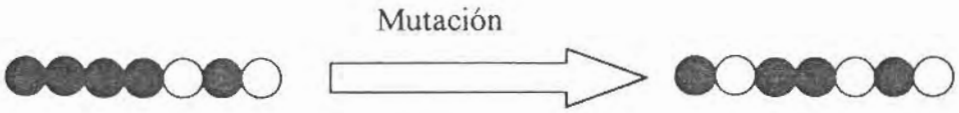


Figura 3.10. Mutación en el segundo bit.

El cromosoma presenta una mutación que se hace en la posición número dos del mismo. La finalidad de una mutación es impedir que las poblaciones se vuelvan homogéneas y la de mantener la diversidad, permitiendo con ello que el proceso evolutivo continúe avanzando.

La mutación se implementa cuando la evaluación de los miembros de la población es muy similar en todos ellos.

3.3.6 La Extinción.

La extinción es una nueva opción propuesto por Sergio Vázquez y Montiel [3], la cual consiste en borrar toda la población después de un cierto número de mutaciones. Al borrar toda la población se conserva el cromosoma mejor evaluado para mantener la memoria de las generaciones anteriores, haciendo una analogía, es comparable con la extinción de los dinosaurios, los cuales tuvieron que dejar el camino libre a otras especies al no poderse adaptar a los cambios del medio ambiente que se presentaron en su época.

3.4 Esquemas o bloques construidos.

En los algoritmos genéticos, la búsqueda de soluciones óptimas de un problema, consiste en la búsqueda de determinadas cadenas binarias. Supongamos un paisaje imaginario, el cual corresponde al universo de todas las cadenas, en este paisaje, la ubicación de las cadenas esta descrita por las cimas y valles de dicho paisaje. Las cimas corresponden a las cadenas más buenas, es decir, la cima que tenga el punto más elevado, ocupa la cadena más óptima, mientras que los valles corresponden a las soluciones menos buenas.

También es posible definir regiones del espacio de soluciones fijándose en las cadenas que posean unos o ceros en lugares determinados, una especie de equivalencia binaria de las coordenadas de un mapa. El conjunto de todas las cadenas que tengan como primer bit, un cero, representan una región en el espacio de posibilidades, así mismo las que tengan como primer bit un uno, o las que tengan un cero en el bit número cinco, etc.

La técnica de exploración de dicho paisaje es la "escalada": se comienza en un punto elegido al azar; si una pequeña modificación mejora la calidad de la solución, se continúa en esa dirección, de no suceder esto, se toma la dirección contraria. Sin embargo, los problemas muy complejos hacen que se tengan paisajes con muchas cimas. Al aumentar el número de dimensiones del problema, el paisaje puede contener además de cimas y valles, puentes y túneles. El encontrar la cima adecuada, e incluso la determinación del sentido del ascenso, se tornan cada vez más problemáticos. Al mismo tiempo que los espacios de búsqueda se vuelven enormes.[16]

Los algoritmos genéticos, lanzan una red sobre el paisaje. Las cadenas de la población que se encuentran dentro de la red, sondean muchas regiones a la vez. La tasa que el algoritmo genético toma muestras en diferentes regiones corresponde directamente con su elevación media, es decir, con la probabilidad de encontrar una buena solución en ese entorno.

Los algoritmos genéticos explotan las regiones de más alto rendimiento del espacio de soluciones ya que las generaciones sucesivas de reproducción y cruzamiento, generan un número creciente de cadenas pertenecientes a ellas. De hecho, el número de cadenas de una región dada, aumenta de manera proporcional a la estimación estadística de la idoneidad de esa región. Una persona dedicada a la estadística, tendría que evaluar docenas de muestras tomadas de millones de regiones para determinar la idoneidad media de cada región. El algoritmo genético, alcanza el mismo resultado con muchísimas menos cadenas y prácticamente si computo alguno.

La parte central de este hecho, reside en el hecho de que cada cadena individual, pertenece a todas las regiones en las cuales aparece uno de cualquiera de sus bits. La cadena 00100110 es miembro de todas las regiones 00****** (el asterisco * indica que el valor del bit correspondiente es indiferente), 0******0, **1**11* y demás. A tales regiones se les llama ***Bloques Construidos o Esquemas*** [3]. Las regiones más amplias, las

que contienen muchos bits sin especificar, serán muestreadas por una fracción grande de todas las cadenas de la población. Así que, un algoritmo genético que manipule una población de varios cientos de cadenas está realmente tomando muestras de un número de regiones enorme. Tal paralelismo implícito proporciona al algoritmo genético su ventaja principal sobre otros procedimientos.

3.5 Conclusiones.

Los algoritmos genéticos, son algoritmos de búsqueda que permiten la exploración en un abanico mucho más amplio de posibles soluciones a diversos tipos de problemas, que los programas tradicionales, su mecanismo de funcionamiento esta basado en la imitación de algunos procesos observados en la evolución natural. Sus principios se fundamentan en la "*teoría de la evolución*" propuesta por ***Charles Darwin***.

Entre sus componentes básicos se encuentran: Generar una población inicial, el módulo de evaluación, de selección, de cruza y el de mutación. Donde cada uno de estos componentes juega un papel fundamental en el funcionamiento del algoritmo.

En el presente capítulo se presentaron las ventajas que presentan estos algoritmos sobre los métodos tradicionales de optimización, así como las consideraciones que se deben tener en cuenta para aplicar un algoritmo genético a un problema.

Finalmente, se da una explicación muy general de porque trabajan estos algoritmos mediante los esquemas o bloques construidos.

CAPITULO IV

IMPLANTACIÓN DEL ALGORITMO

Introducción.

Una vez que se evaluaron las técnicas de optimización, se vio que la mayoría tiene una fuerte dependencia del punto inicial donde se comienza la búsqueda de la solución.

Se observó que los algoritmos genéticos presentan mayor robustes y que la búsqueda de soluciones es paralela, permitiendo así, la exploración en un abanico mucho más grande del espacio de solución que los métodos tradicionales.

Debido a estas características, se eligió para obtener la forma de una superficie asférica que mejor se ajusta a una distribución de puntos la técnica de algoritmos genético en la fase de optimización. Se utilizó para la implantación del mismo la herramienta C++ de Borland.

En el presente capítulo se presenta la forma en que fue implementado el algoritmo, se presentan los diagramas de flujo de todas las partes integrantes del mismo. En el anexo 1 se presenta el código completo.

4.1 Programa principal.

El programa principal cuenta con las llamadas a las diferentes funciones que componen al algoritmo genético, la figura 4.1 presenta el diagrama de flujo correspondiente al programa principal.

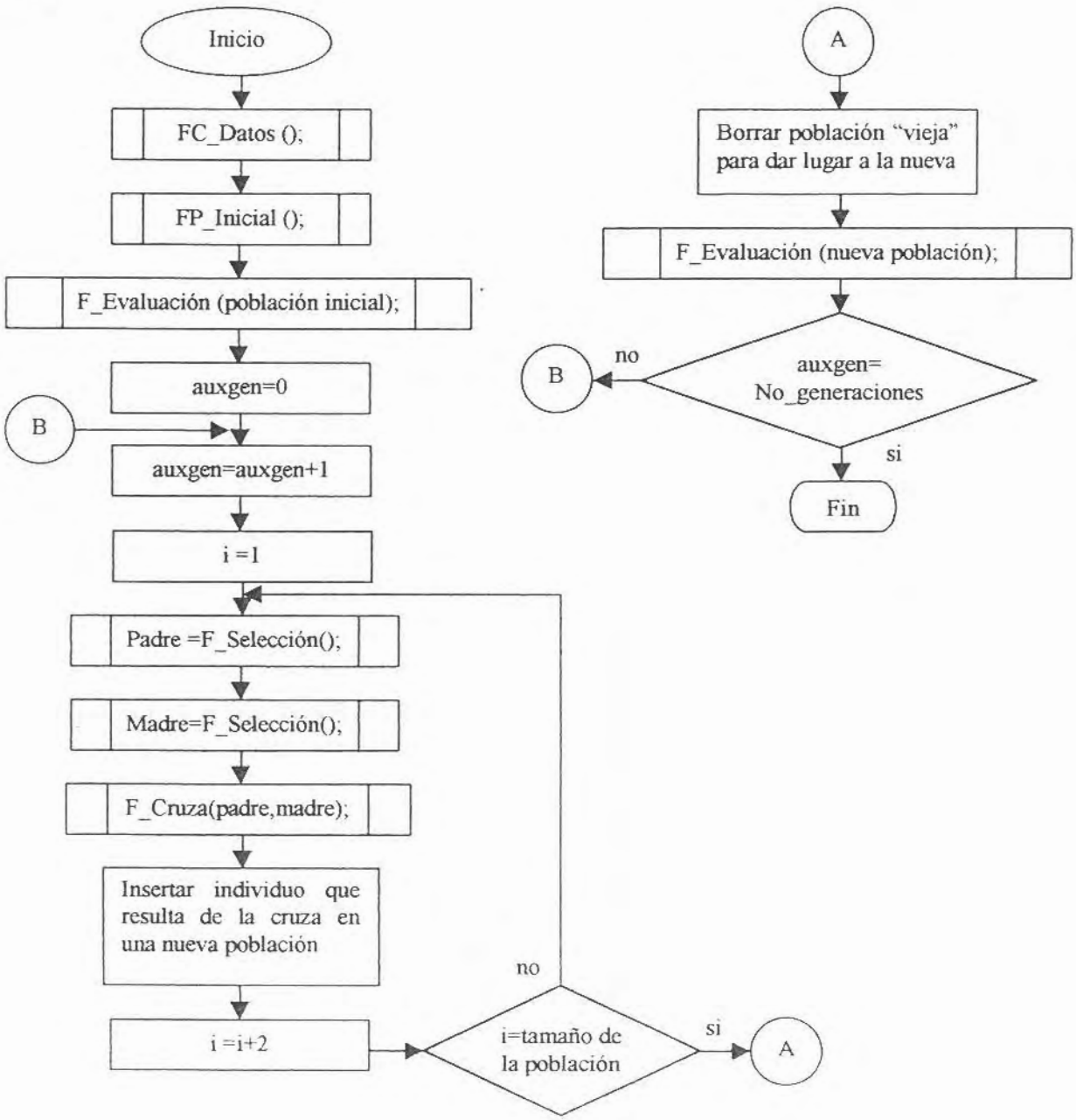


Figura 4.1. Diagrama de flujo del programa principal.

El programa principal, como primer paso, llama a la función **FC_Datos();** la cual se encarga de establecer los parámetros bajo los cuales trabajará el algoritmo, ya que en esta se especifica el tamaño de la población (*Tpob*), la longitud del cromosoma o individuo (*Lcromo*) y finalmente el número de generaciones (*No_generaciones*).

Como segundo paso, se llama a la función **FP_Inicial()**;, la cual es la que se encarga de generar la población inicial, dicha población será la que se evaluará como primera instancia. Una vez generada la población inicial, se manda a evaluar, la función **F_Evaluación**(*población inicial*); es la encargada de realizar dicha acción.

Posteriormente, se seleccionan dos cromosomas mediante la función **F_Selección**(); la cual regresa el número de cromosoma seleccionado, una vez seleccionados se procede a realizar la cruza de los mismos a través de **F_Cruza**(*cromosoma1, cromosoma2*);. La función de cruza tiene como resultado dos cromosomas nuevos, los cuales son miembros de la nueva población. Este proceso se realiza hasta que el número de individuos de la nueva población sea igual al tamaño de la población definido en **FC_Datos**();.

Finalmente se borran los individuos "viejos", para dar lugar a los nuevos individuos, los cuales son evaluados por **F_Evaluación**(*nueva población*);. El algoritmo termina cuando se cubran el número de generaciones definido en **FC_Datos**();.

En las siguientes secciones se explican más a detalle las diferentes funciones antes descritas, presentando los diagramas de flujo correspondientes.

4.2. Función de datos iniciales.

La función **FC_Datos()**; es la que se encarga de pedir los datos iniciales del programa, los cuales son:

- Tamaño de la población (*Tpob*).
- Longitud del cromosoma (*Lcromo*).
- Número de generaciones (*No_generaciones*).

Las palabras entre los paréntesis que aparecen delante de los datos pedidos, corresponden a las variables asociadas dentro del programa.

En esta función es donde de establecen los parámetros bajo los cuales trabajará el algoritmo, es decir, se proporciona el número de cadenas con los que contará la población (*Tpob*), la longitud de cada cromosoma (*Lcromo*), y finalmente el número de generación que se necesitan para detener el algoritmo (*No_generaciones*). Ver figura 4.2.

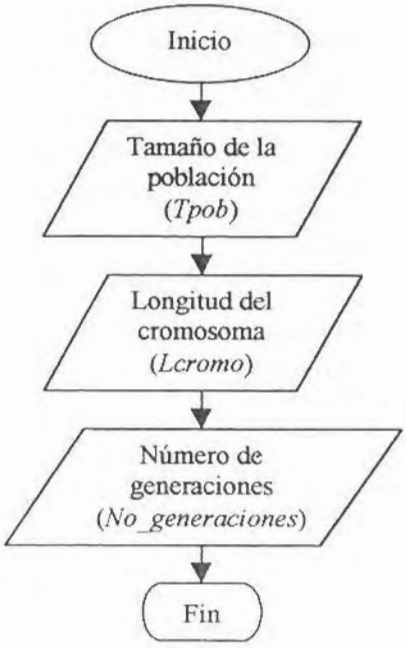


Figura 2.2. Diagrama de flujo de la función FC_Datos();.

4.3 Función para Generar Población Inicial.

La función **FP_Inicial()**, es la que se encarga de crear aleatoriamente la población, para ello se hace uso de una **lista enlazada**, la cual es una colección lineal de *estructuras autorreferenciadas.* [21]

Las estructuras son colecciones de variables relacionadas. Una estructura puede contener variables de diferentes tipos de datos, es decir puede contener variables tipo *char, int, float*, etc.

Una estructura autorreferenciada, contiene un miembro de apuntador que "apunta" a una estructura al mismo tipo de estructura. Ver figura 4.3;

```
struct lista {
        int UTM;
        struct lista *sig_ptr;
            }
```

Figura 4.3. Definición de una estructura.

El figura 4.3 define una estructura que tiene dos miembros, uno es el entero *UTM*, y el otro es el apuntador *sig_ptr*. El miembro *sig_ptr* apunta a una estructura del tipo *struct lista*, es decir el miembro *sig_ptr* se utiliza para "vincular" una estructura del tipo *struct lista* con otra estructura del mismo tipo.

La figura 4.4 muestra un ejemplo de tres estructuras autorreferenciadas, enlazadas juntas para formar una lista ligada. El tercer miembro de la lista presenta una diagonal, la cual representa al apuntador NULL para indicar que el enlace no apunta a otra estructura.

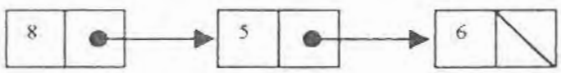


Figura 4.4. Tres estructuras autorreferenciadas, enlazadas.

Las listas enlazadas presentan varias características, una de ellas y por la cual se decidió ocuparlas, es que son apropiadas cuando no es prescindible de inmediato el número de datos a representarse en la estructura, son dinámicas, es decir, su tamaño "crece" y se "encoge" en tiempo de ejecución. Esta característica de las listas hace que la memoria del sistema se utilice de manera óptima, situación que no sucedería si se ocuparan arreglos, ya que el tamaño del arreglo no puede ser modificado, debido a que la memoria de este es asignada en tiempo de compilación.

La lista *Lista_pob*, figura 4.5 es utilizada para almacenar a los cromosomas de la población generados aleatoriamente, en un principio la inicial, y subsecuentemente las poblaciones que se van generando. Cuenta con dos miembros, el miembro *unsigned char Cromo*, y el miembro apuntador *next_nodo*.

```
struct Lista_pob{
        unsigned char Cromo;
        struct Lista_pob *next_nodo;}
```

Figura 4.5. Estructura para almacenar los cromosomas; *Lista_pob*.

La forma en que se representan los cromosomas es por medio de cadenas de bits, una secuencia de 8 bits forma un byte, un byte es la unidad estándar de almacenamiento para una variable tipo *char*[21], ya que esta tiene una longitud de 8 bits, el problema es que su rango es de –128 a 127, por lo que se hace uso del modificador *unsigned* el cual no toma en cuenta el signo. Entonces la variable *Cromo*, que es del tipo *unsigned char*, tiene una longitud de 8 bits (1 byte) y un rango de 0 a 255.

Una situación muy importante, es la conocer el número de bits que serán utilizados para así, posteriormente, conocer el número de estructuras o nodos que contendrá la lista, para ello se utiliza la variable *int TotalBits*, la cual tendrá como resultado el producto de *Tpob* y *Lcromo*.

Ejemplo (4.1). si se tiene una población de 100 individuos y la longitud de cada uno de ellos es de 25, entonces el total de bits (*TotalBits*) que se manejarán es de 2500.

A continuación se procede a determinar el número de nodos, ya que para cada nodo corresponde una estructura perteneciente a la lista. Para ello se divide el número total de bits entre 8, como se mencionó con anterioridad, la lista *Lista_pob*, cuenta con un miembro *unsigned char Cromo*, el cual es de 8 bits, o un byte. Por esta razón, es necesario saber cuantos bytes se necesitarán para así formar la población inicial.

La variable *NoNodos*, es la que se encarga de guardar este resultado, retomando el ejemplo anterior (4.1): se tienen 2500 bits, entonces el número de nodos es igual a 2500/8, lo que da un resultado de 312.5. Cuando se tiene como resultado de esta división, números no enteros, lo que se hace es incrementar en una unidad el número de bytes, para así tener un número entero de bytes, por lo que la variable *NoNodos* tendrá entonces, 313 bytes. Esto quiere decir entonces, que la lista *Lista_pob* tendrá 313 estructuras ligadas.

Una vez teniendo estos valores, se procede a generar la población inicial, para ello, se hace uso de un ciclo *for*, el cual irá desde 1 hasta el número de nodos especificados por *NoNodos*, para el ejemplo que ha seguido hasta este momento, 313.

Lo primero que se hace es llamar a la función **srand((unsigned)time(&t));** de la biblioteca estándar de C, la cual toma un argumento entero (semilla), para que en cada ejecución del programa, la función **rand()** produzca una secuencia diferente de números aleatorios, la "semilla" **(unsigned)time(&t)** hace que la computadora lea su reloj para obtener de manera automática un valor para dicha "semilla".

Se hace uso del operador de módulo (%), para producir números enteros en el rango de 0 a 255, con esto, se garantiza que se obtendrá un número decimal que tenga su equivalente en código ASCII.

Una vez teniendo este número aleatorio, su equivalente en código ASCII, se guardará en la variable *Cromo* de la primera estructura de la lista, posteriormente, se crea un nuevo nodo y finalmente se apunta al siguiente nodo (estructura), para almacenar en este al siguiente caracter obtenido también aleatoriamente, el proceso continúa hasta que el ciclo *for* termine.

Figura 4.6: Supóngase que la función **rand()** devuelve los valores 64, 35 y 115 para las primeras tres estructuras y el valor 91 para la última estructura de las 313 que se tienen dentro de la lista, sus valores correspondientes en código ASCII son, para las primeras tres, **@**, **#**, **s**, respectivamente, y **[** para la última, estos valores quedarán en la lista de la siguiente manera

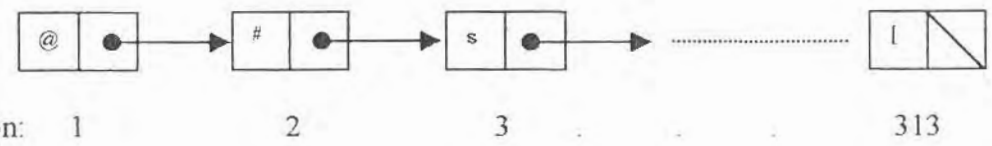


Figura 4.6. Lista enlazada que muestra los valores almacenados en las diferentes estructuras que la conforman.

Cuando el ciclo acabe, la lista *Lista_pob*, tendrá ya almacenados a todos los miembros de la población inicial. Finalmente cuando se sale del ciclo, la última estructura de la lista, se "aterriza" a **NULL**, para indicar que esta estructura, no apunta a otra.

A continuación se presenta el diagrama de flujo para la función **FP_Inicial();**

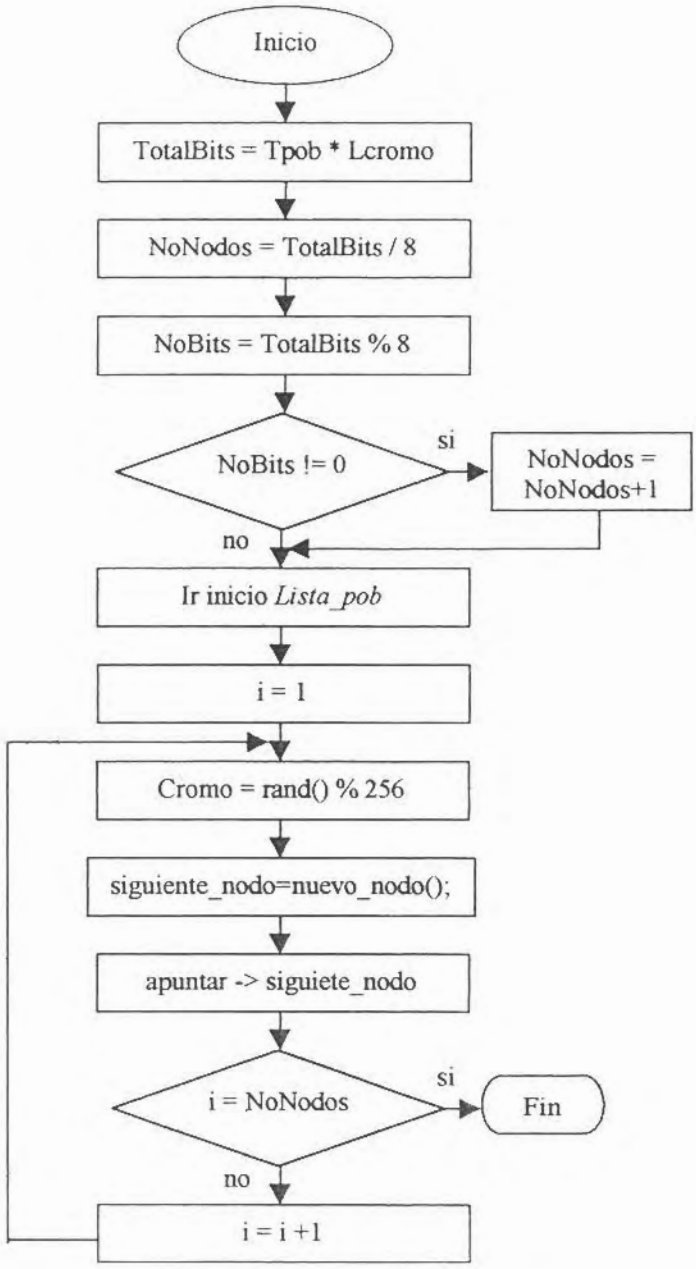


Figura 4.7. Diagrama de Flujo para la Función FP_Inicial();.

4.4 Función de Evaluación.

La función **F_Evaluación()**, es la que se encarga de evaluar la aptitud de cada cromosoma de la población, recibe como parámetro el cromosoma que será evaluado.

La función a optimizar esta dada por

$$f_i = (R_{ie} - R_{iT})^2, \tag{4.1}$$

donde R_{ie} es una constante que se obtiene por medio de un archivo (rad20.dat) el cual contiene los datos X, Y y R_{ie}, correspondientes a radios de curvatura para superficies asféricas; R_{iT}, es la que obtiene el algoritmo, y esta definida por

$$R_{iT} = \frac{1}{C_0}\left(1 - K_0 C_0^2 S^2\right)^{1/2}, \tag{4.2}$$

donde

$$S = \sqrt{X^2 + Y^2}, \tag{4.3}$$

K_0 y C_0 son las variables que genera el algoritmo. Los valores de X y Y, se obtienen del mismo archivo (rad20.dat). Los datos del archivo se muestran en el anexo 2.

R_{iT} tiene dos variables desconocidas, las cuales son C_0 y K_0, variables que se codificarán. Por esta razón, la longitud del cromosoma (*Lcromo*) debe ser un número que sea múltiplo de dos como 8, 12, 32, etc. Como se mencionó en el apartado 4.3, la lista *Lista_pob* contiene todos los miembros que serán evaluados, supongamos que la longitud del cromosoma es de 16 bits, entonces, para formarlo, serán necesarios dos miembros de la lista *Lista_pob*, ya que cada uno de ellos es de 8 bits, si el cromosoma es de 40 bits, entonces serán necesarios cinco miembros de la lista.

Para la codificación del cromosoma, es necesario "partirlo" en dos partes, en donde cada una de ellas corresponde a una de las dos variables que se están codificando. Las variables pueden tomar cualquier valor dentro de un rango especifico, los cuales están delimitados por

$K = [-4,4]$

$C = [0,0.5]$

Para que la codificación tenga efecto, se hace un "mapéo" para cada una de las variable, el cual se presenta a continuación.

$$\left.\begin{aligned} C_0 &= \frac{\text{Valor decimal de la parte que le corresponde del cromosoma.}}{(2^{\text{Longitud del cromosoma}/2}-1)*200} \\ K_0 &= 4 - \frac{\text{Valor decimal de la parte que le corresponde del cromosoma} * 8}{(2^{\text{Longitud del cromosoma}/2}-1)} \end{aligned}\right\} \quad (4.4)$$

Una vez decodificados estos parámetros, se procede a evaluar esta solución, encontrando para ello, los radios R_{iT} para cada punto de la superficie dado en el archivo (asfer.dat) y se realiza lo siguiente

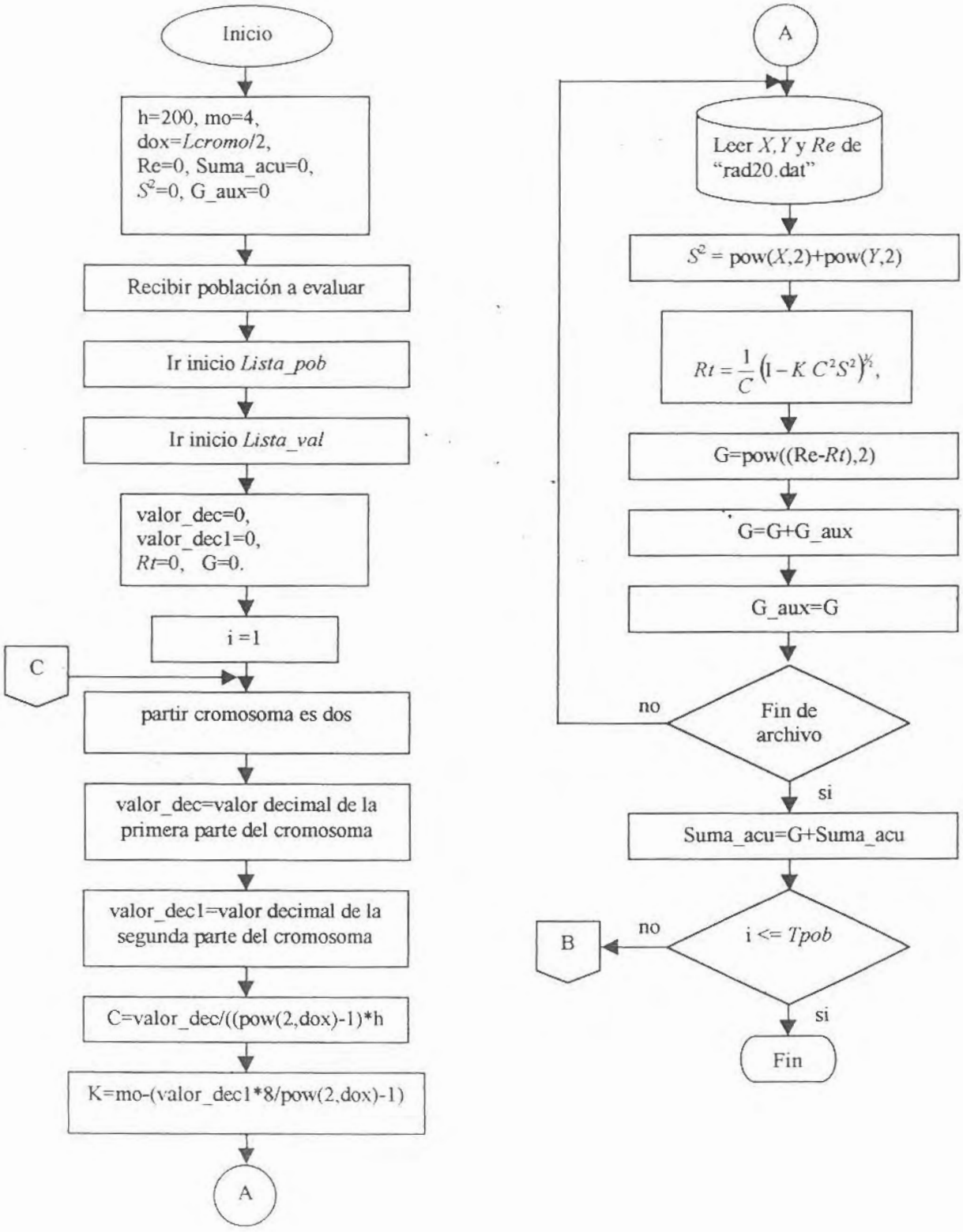
$$f(i)=\sum_{i=1}^{N}(R_{ie}-R_{iT}), \quad (4.5)$$

Donde N es el número de datos que contiene el archivo (asfer.dat), el resultado de la expresión 4.5 es la evaluación de un cromosoma con cada uno de los datos del archivo (asfer.dat). Este resultado se guarda, con la finalidad de ir calculando la suma acumulativa de la evaluación de todos los cromosomas, ya que este proceso se efectúa para todos los miembros de la población. La función de evaluación recibe como parámetro la población que se evaluará. Los valores de la función de evaluación son almacenados, en una lista autorreferenciada llamada *Lista_val*, la cual se presenta a continuación:

```
struct Lista_val{
     float valor_dec;
     float valor_dec1;
     double RT;
     double G;
     struct Lista_val *next_nodo;
                    }
```

Figura 4.8. Estructura para almacenar los cromosomas; *Lista_val.*

A continuación se presenta el diagrama de flujo para la función **F_Evaluación();**


Inicio
h=200, mo=4,
dox=Lcromo/2,
Re=0, Suma_acu=0,
S²=0, G_aux=0
Recibir población a evaluar
Ir inicio Lista_pob
Ir inicio Lista_val
valor_dec=0,
valor_dec1=0,
Rt=0, G=0.
i =1
C
partir cromosoma es dos
valor_dec=valor decimal de la primera parte del cromosoma
valor_dec1=valor decimal de la segunda parte del cromosoma
C=valor_dec/((pow(2,dox)-1)*h
K=mo-(valor_dec1*8/pow(2,dox)-1)
A
A
Leer X, Y y Re de "rad20.dat"
S² = pow(X,2)+pow(Y,2)
Rt = 1/C (1 - K C²S²)^½,
G=pow((Re-Rt),2)
G=G+G_aux
G_aux=G
no
Fin de archivo
si
Suma_acu=G+Suma_acu
B
no
i <= Tpob
si
Fin

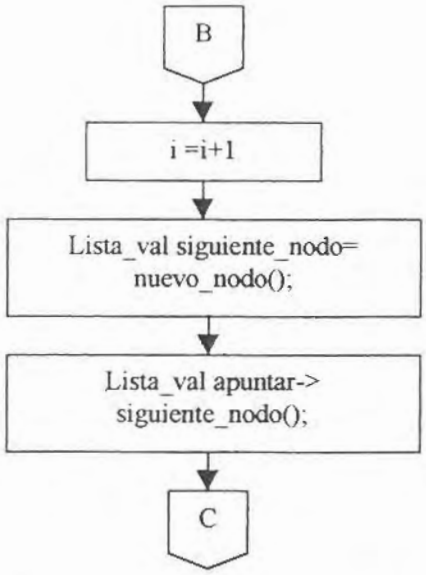


Figura 4.9. Diagrama de flujo de la función F_Evaluación();.

La *Lista_val* contiene las evaluaciones de todos los cromosomas involucrados en el proceso.

Función de Selección.

La función **F_Selección()**, es la encargada de seleccionar a los cromosomas que se cruzarán para formar la siguiente generación. Se implantó la técnica de la rueda de la ruleta, vista en el capítulo 3, para llevar a cabo dicha selección.

A continuación se muestra el diagrama de flujo de dicha función. Ver figura 4.10. La variable *Suma_acu*, contiene la suma acumulativa de las evaluaciones hechas en la función de evaluación, *pastel* contiene un número aleatorio entre 0 y la suma acumulativa, *i_cromo* corresponde al número de cromosoma que será devuelto por la función de selección. *G*, corresponde a la evaluación del cromosoma que se está evaluando.

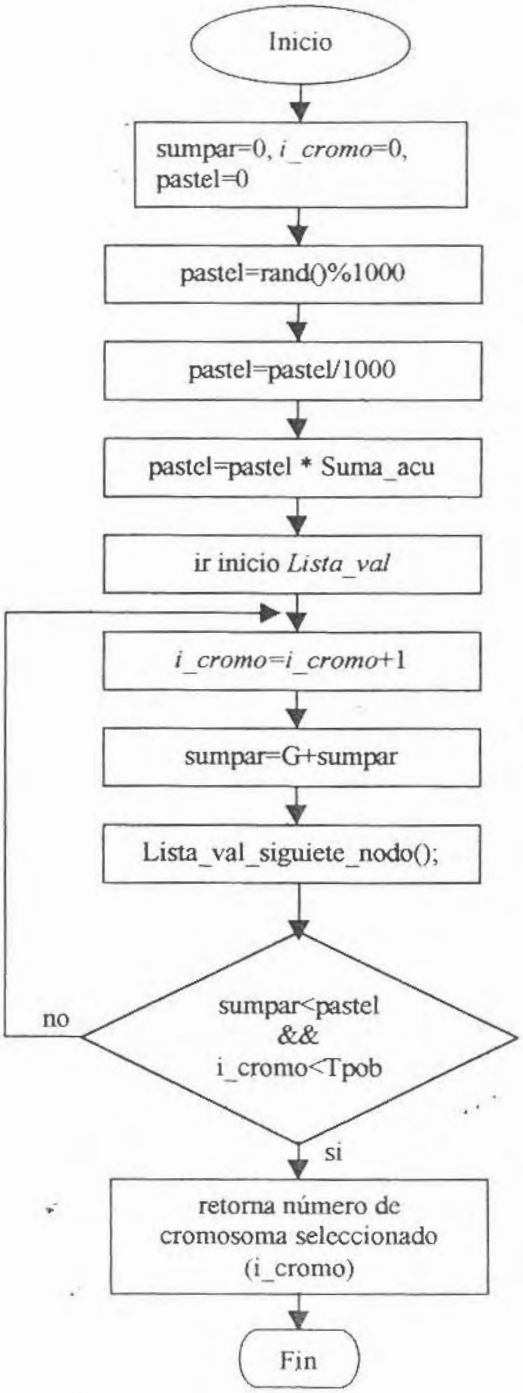


Figura 4.10. Diagrama de flujo de la función F_Selección();.

4.6 Función de Cruza.

La función **F_Cruza()**, recibe como parámetros dos números enteros, los cuales corresponden a los dos cromosomas (*padre* y *madre*) seleccionados por la función de selección, se seleccionó la cruza en un punto descrita en el capítulo 3, para llevar a cabo dicha función.

La variable *Punto_cruza*, contiene un número aleatorio entre 1 y la longitud del cromosoma (*Lcromo*) para que en ese punto aleatorio se realice la cruza.

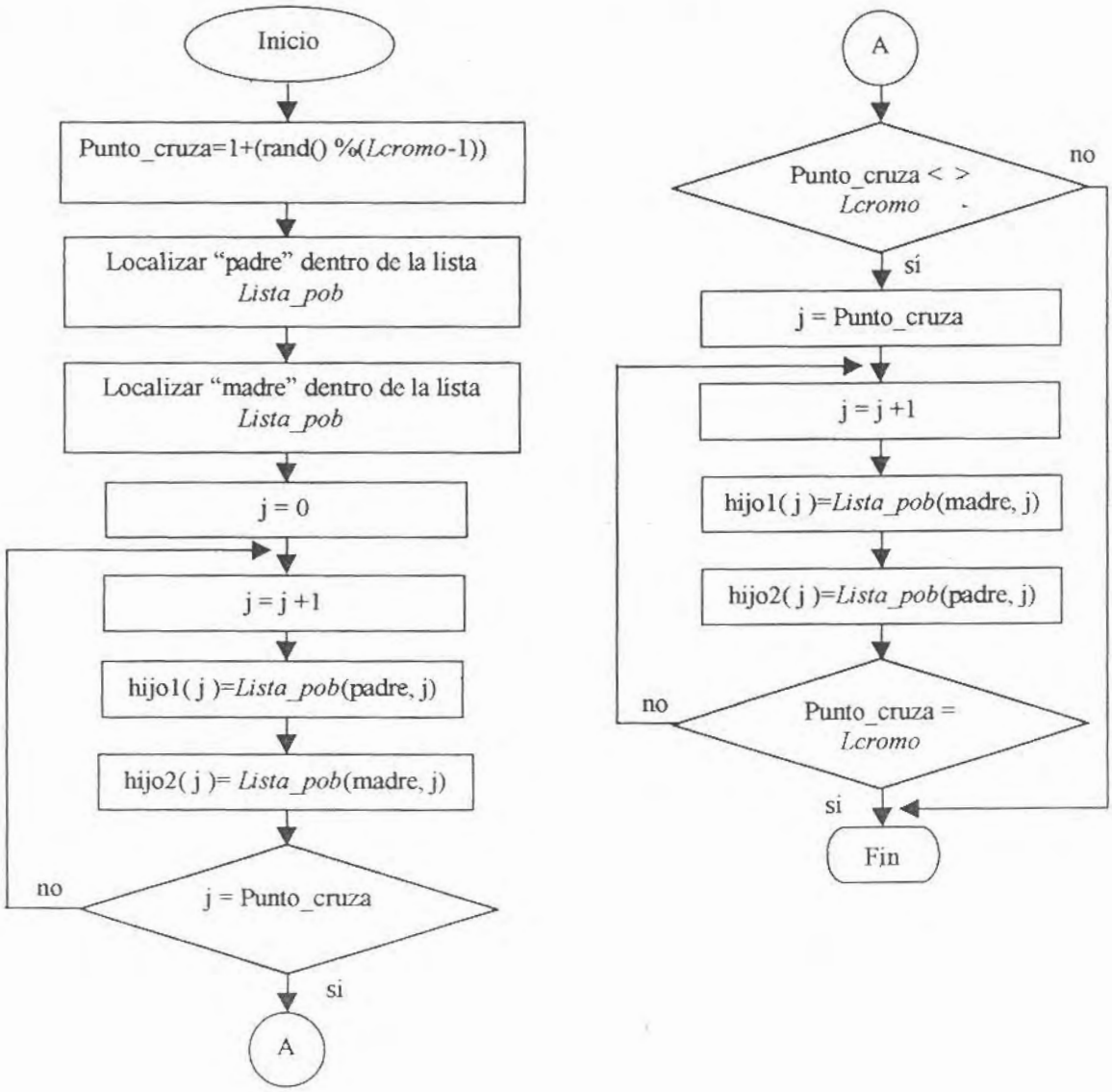


Figura 4.11. Diagrama de flujo de la función F_Cruza();.

Para que los resultados del algoritmo se pudieran visualizar, se realizó una función la cual grafica la superficie obtenida, esta función recibe los parámetros de C y K obtenidos por el algoritmo, como las superficies que se sometieron a prueba tienen un diámetro de 200, entonces, en el plano tridimensional, x va de -100 a 100, al igual que y, z esta definida por

$$z = \frac{CS^2}{1+\left(1-(K+1)C^2S^2\right)^{1/2}}, \tag{4.6}$$

La función **Gafica()**;, se presenta en el anexo 1.

BIBLIOTE

CAPÍTULO V

PRUEBAS Y RESULTADOS

Introducción.

Para comprobar la eficiencia del algoritmo, se realizaron varias pruebas, la primera consta de tres casos, en el primero se mantiene el tamaño de la población constante, variando la longitud del cromosoma y el número de generaciones.

En el segundo caso, se mantiene la longitud del cromosoma constante y se varia el tamaño de la población y el número de generaciones. Finalmente, para el tercer caso, el tamaño de la población y la longitud del cromosoma se varían y el número de generaciones es constante. Para esta primer prueba, se realizaron 192 corridas del algoritmo. El archivo de datos del cuál se leen los valores de X, Y, y Re, es el archivo "rad20.dat", el cual fue construido para valores de C = 0.005, y K = -2.00, por lo tanto estos son los valores óptimos que debe regresar el algoritmo.

En las pruebas 2 y 3 se prueba el algoritmo con diferentes archivos de datos, en la prueba 4, se realiza una reducción del espacio de búsqueda de la solución, finalmente en la prueba 5 se agrega un coeficiente de deformación.

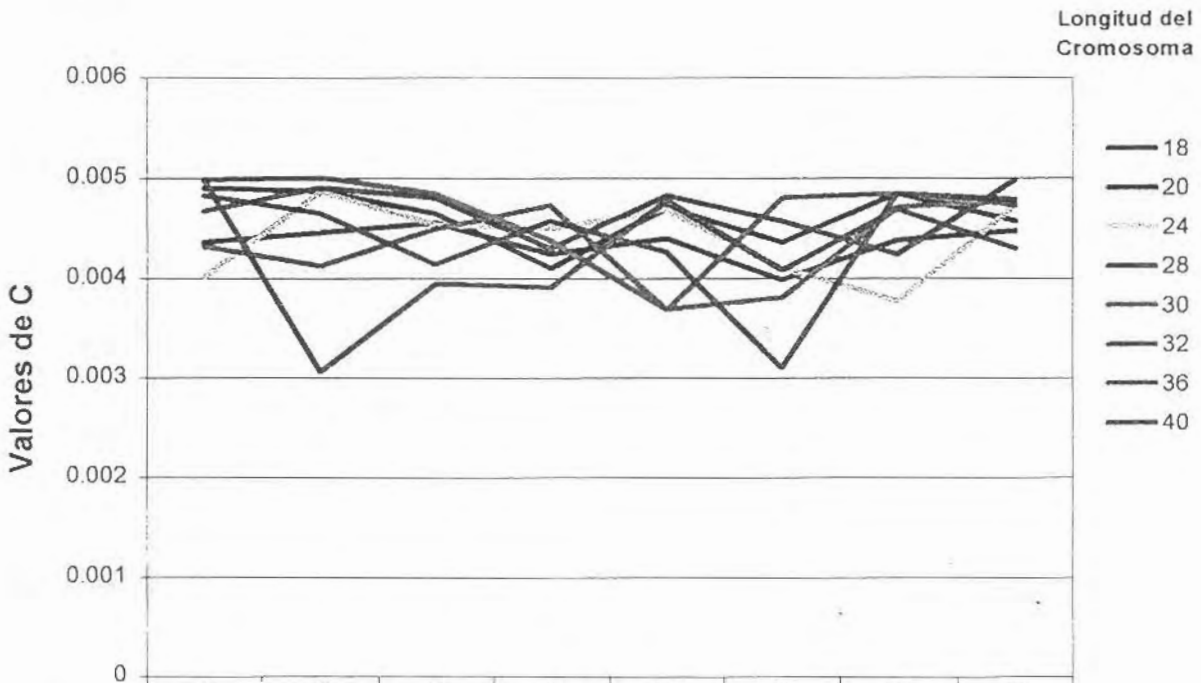
Estas pruebas se realizan con el objetivo de verificar el funcionamiento del algoritmo, y bajo que parámetros (tamaño de la población, longitud del cromosoma y número de generaciones) funciona de manera óptima.

5.1 Prueba Número 1

5.1.1 Caso 1

Para el caso número 1, el tamaño de la población es constante y corresponde a 120 individuos, posteriormente se varía la longitud del cromosoma en 18, 20, 24, 28, 30, 32, 36 y 40 cromosomas, finalmente se varía el número de generaciones en 8, 10, 16, 20, 25, 30, 32 y 38.

La gráfica 5.1, muestra los valores obtenidos de C por el algoritmo para el caso 1.



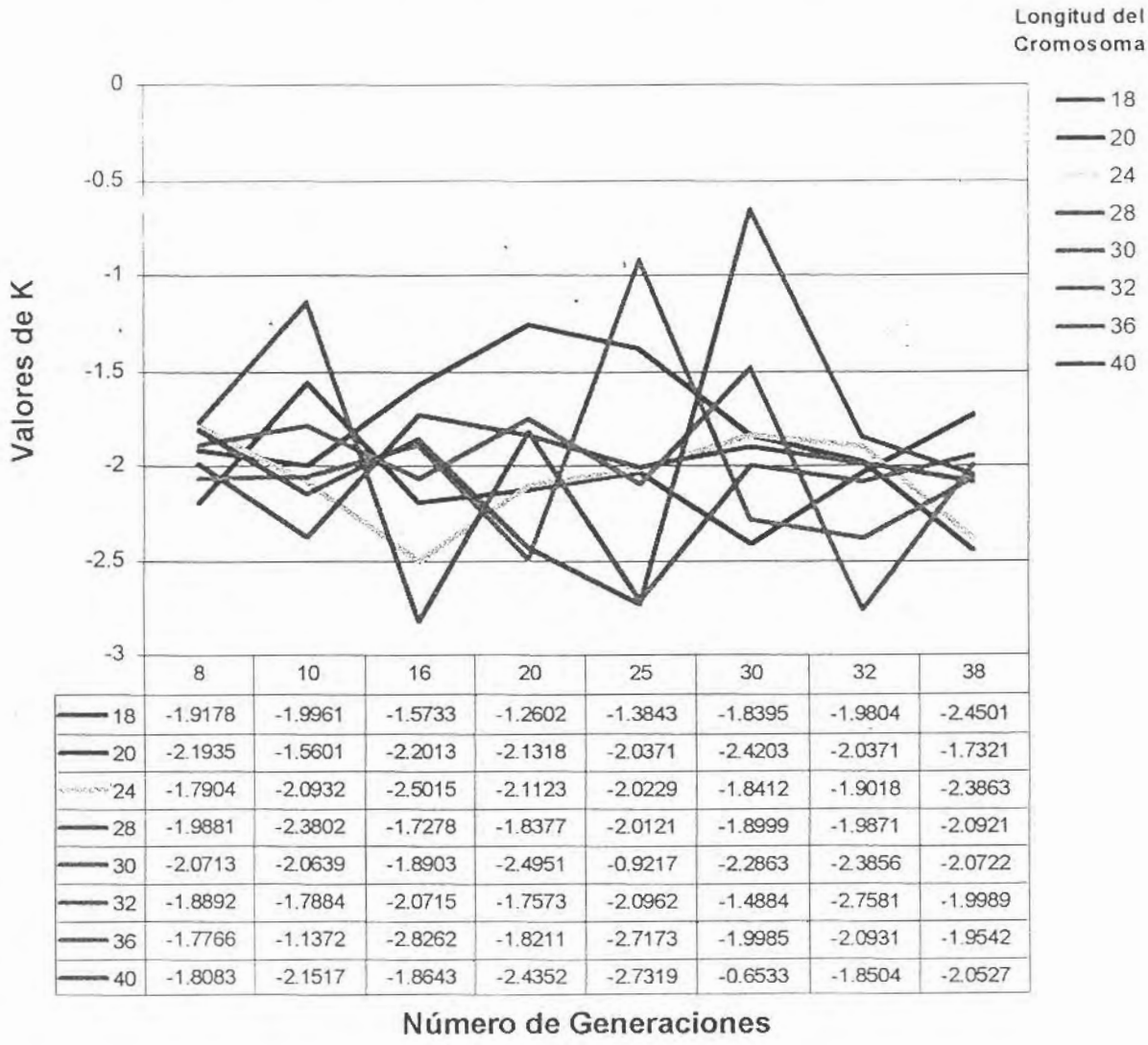
	8	10	16	20	25	30	32	38
18	0.004354	0.004442	0.004551	0.004237	0.004393	0.003982	0.004374	0.004462
20	0.004902	0.004853	0.004653	0.004101	0.004697	0.004361	0.004834	0.004567
24	0.004016	0.004861	0.004524	0.004491	0.004691	0.004105	0.003756	0.004724
28	0.004675	0.004897	0.004809	0.004301	0.004829	0.004562	0.004235	0.004973
30	0.004321	0.004118	0.004492	0.004724	0.003683	0.003808	0.004712	0.004771
32	0.004972	0.004991	0.004844	0.004364	0.003684	0.004801	0.004848	0.004721
36	0.004983	0.003063	0.003942	0.003906	0.004778	0.004081	0.004682	0.004297
40	0.004819	0.004644	0.004131	0.004571	0.004264	0.003097	0.004848	0.004791

Número de Generaciones

Gráfica 5.1. Valores de C obtenidos por el algoritmo.

Como se puede observar existe muchos valores cercanos al óptimo, la mayoría se encuentran ubicados entre 0.004 y 0.005, entre los más cercanos se observan los valores de 0.004983, 0.004991 y 0.004972.

La gráfica 5.2, muestra los valores de *K* obtenidos por el algoritmo para el caso 1.



	8	10	16	20	25	30	32	38
18	-1.9178	-1.9961	-1.5733	-1.2602	-1.3843	-1.8395	-1.9804	-2.4501
20	-2.1935	-1.5601	-2.2013	-2.1318	-2.0371	-2.4203	-2.0371	-1.7321
24	-1.7904	-2.0932	-2.5015	-2.1123	-2.0229	-1.8412	-1.9018	-2.3863
28	-1.9881	-2.3802	-1.7278	-1.8377	-2.0121	-1.8999	-1.9871	-2.0921
30	-2.0713	-2.0639	-1.8903	-2.4951	-0.9217	-2.2863	-2.3856	-2.0722
32	-1.8892	-1.7884	-2.0715	-1.7573	-2.0962	-1.4884	-2.7581	-1.9989
36	-1.7766	-1.1372	-2.8262	-1.8211	-2.7173	-1.9985	-2.0931	-1.9542
40	-1.8083	-2.1517	-1.8643	-2.4352	-2.7319	-0.6533	-1.8504	-2.0527

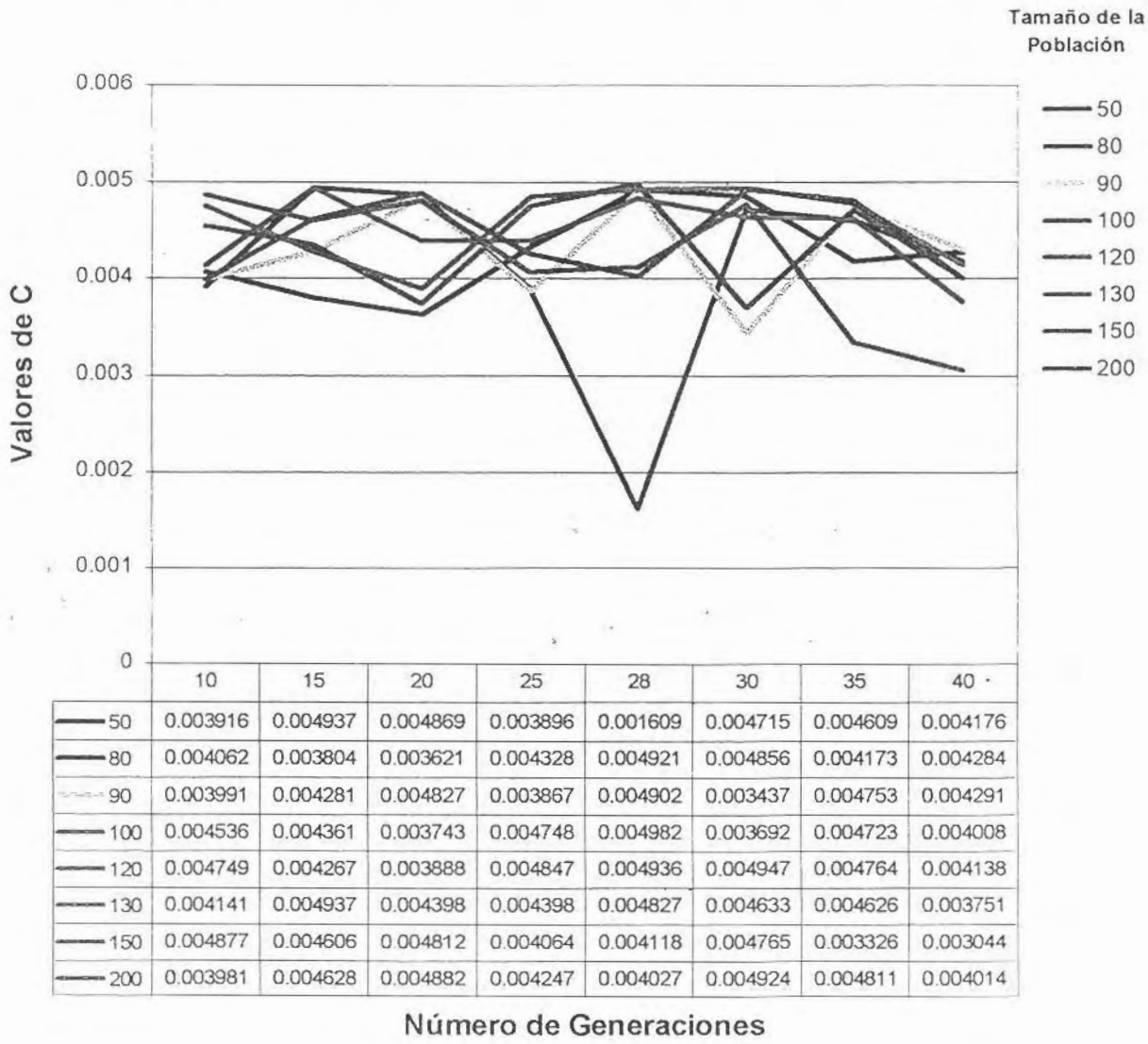
Gráfica 5.2. Valores de *K* obtenidos por el algoritmo.

Al igual que en la gráfica anterior se nota que existen valores cercanos al óptimo, la mayoría de los valores obtenidos por el algoritmo se encuentran entre –1.5 y –2.00

5.1.2 Caso 2

Para el caso número 2, la longitud del cromosoma es constante y es de 32 bits, posteriormente se varía el tamaño de la población en 50, 80, 90, 100, 120, 130, 150 y 200 individuos, finalmente se varía el número de generaciones en 10, 15, 20, 25, 28, 30, 35 y 40.

La gráfica 5.3, muestra los valores de *C* obtenidos por el algoritmo para el caso 2.

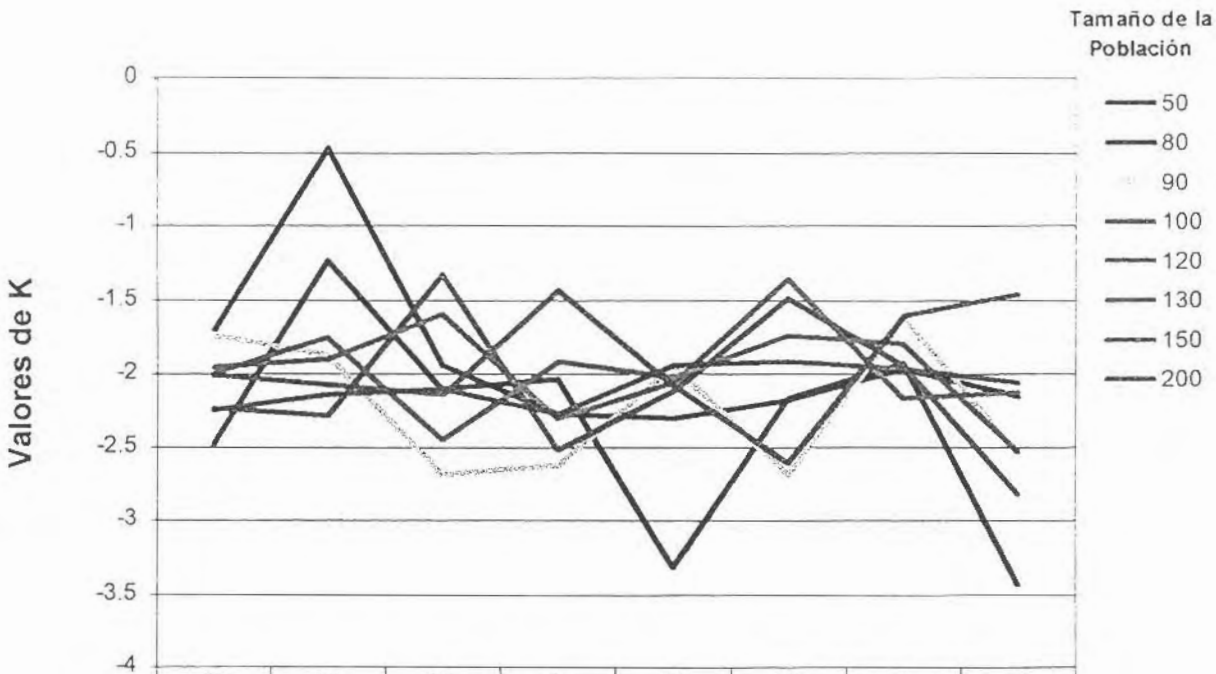


	10	15	20	25	28	30	35	40
50	0.003916	0.004937	0.004869	0.003896	0.001609	0.004715	0.004609	0.004176
80	0.004062	0.003804	0.003621	0.004328	0.004921	0.004856	0.004173	0.004284
90	0.003991	0.004281	0.004827	0.003867	0.004902	0.003437	0.004753	0.004291
100	0.004536	0.004361	0.003743	0.004748	0.004982	0.003692	0.004723	0.004008
120	0.004749	0.004267	0.003888	0.004847	0.004936	0.004947	0.004764	0.004138
130	0.004141	0.004937	0.004398	0.004398	0.004827	0.004633	0.004626	0.003751
150	0.004877	0.004606	0.004812	0.004064	0.004118	0.004765	0.003326	0.003044
200	0.003981	0.004628	0.004882	0.004247	0.004027	0.004924	0.004811	0.004014

Gráfica 5.3. Valores de *C* obtenidos por el algoritmo.

Al igual que en el caso número 1, la mayoría de los valores se encuentran dentro del rango 0.004 y 0.005.

La gráfica 5.4, muestra los valores de K obtenidos por el algoritmo para el caso 2.



	10	15	20	25	28	30	35	40
50	-2.4856	-1.2421	-2.1049	-2.0342	-3.3221	-2.1676	-1.9341	-3.4413
80	-1.7068	-0.4823	-1.9464	-2.2841	-2.3001	-2.1873	-1.9732	-2.1658
90	-1.7481	-1.8842	-2.6974	-2.6207	-1.9401	-2.6884	-1.6229	-2.5418
100	-2.2442	-2.2898	-1.3329	-2.5226	-2.1141	-1.4928	-1.9529	-2.8291
120	-1.9604	-1.9041	-1.5967	-2.3001	-2.0716	-1.3535	-2.1747	-2.1335
130	-1.9978	-1.7609	-2.4563	-1.9176	-2.0254	-1.7451	-1.7991	-2.5359
150	-2.0193	-2.0843	-2.1467	-1.4368	-2.0784	-2.6095	-1.6076	-1.4679
200	-2.2558	-2.1485	-2.1105	-2.2753	-1.9416	-1.9239	-1.9672	-2.0655

Número de Generaciones

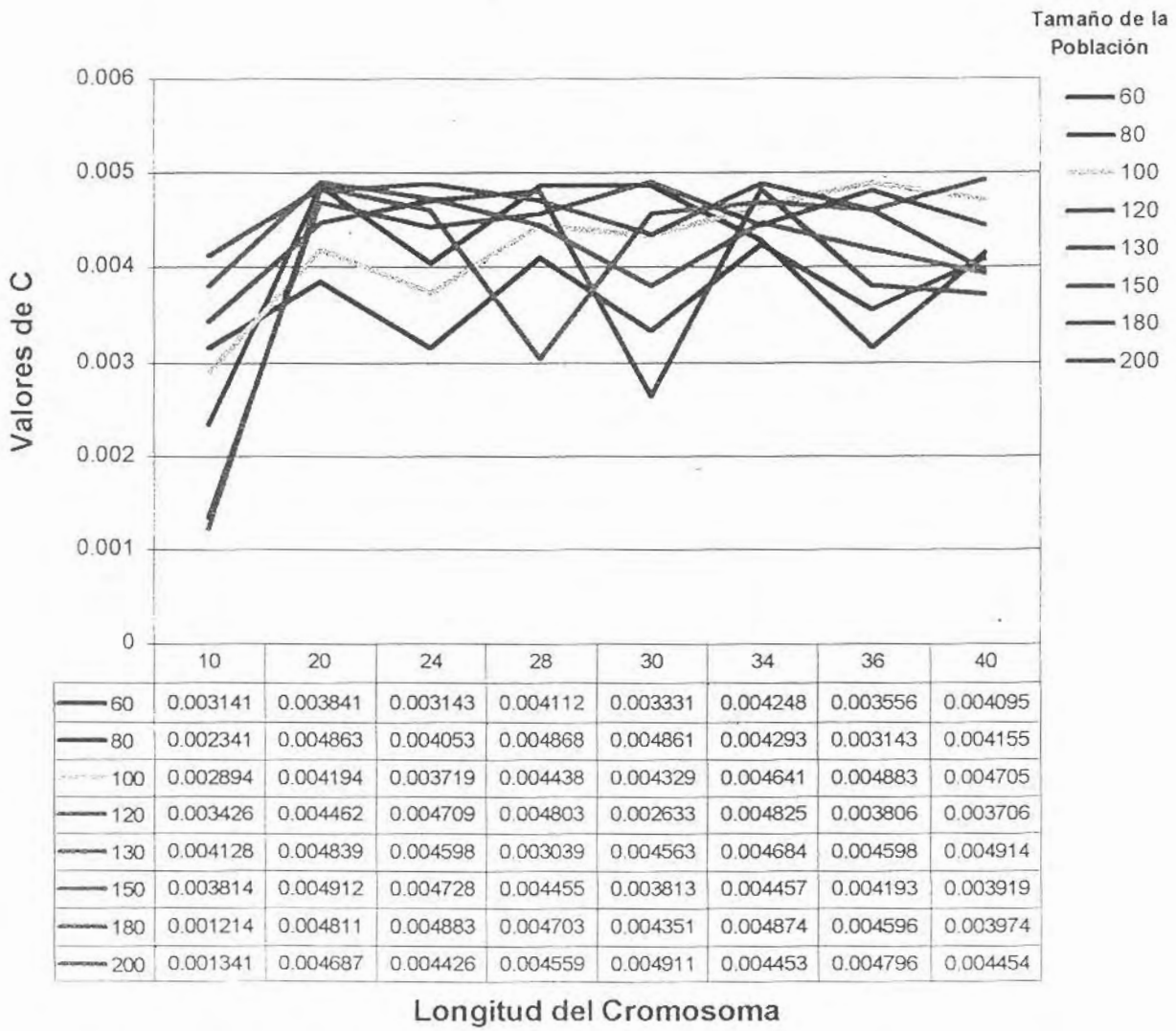
Gráfica 5.4. Valores de K obtenidos por el algoritmo.

Al igual que el caso 1, la mayoría de los valores se encuentran entre –1.5 y -2.00, entre los valores más cercanos al óptimo se encuentran; -1.9978, -2.0193 y –2.0254

5.1.3 Caso 3

Para el caso número 3, el número de generaciones es constante y es de 20 generaciones, posteriormente se varía el tamaño de la población en 60, 80, 100, 120, 130, 150, 180 y 200 individuos, finalmente se varía la longitud del cromosoma en 10, 20, 24, 28, 30, 34, 36 y 40 bits.

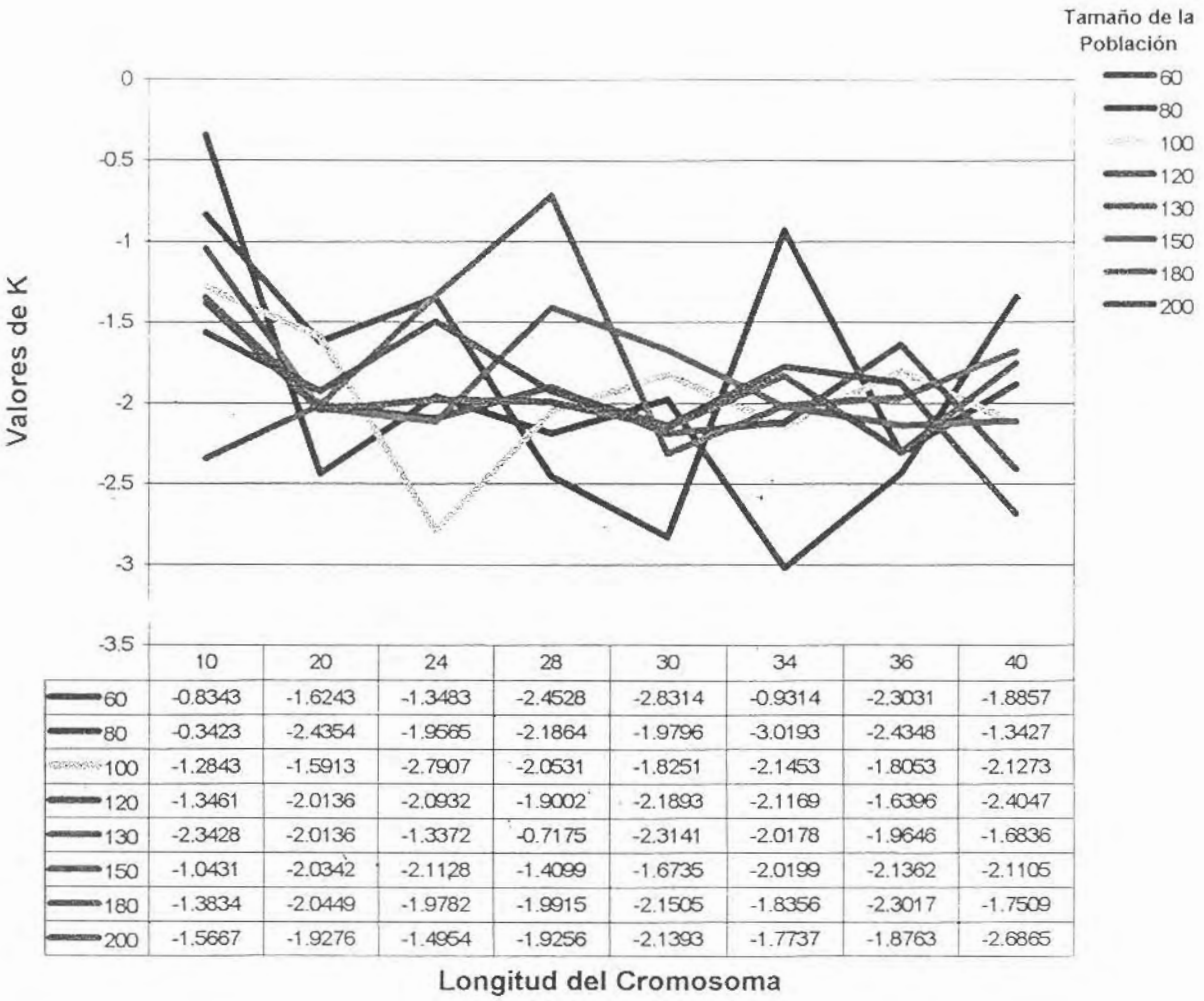
La gráfica 5.5, muestra los valores de *C* obtenidos por el algoritmo para el caso 3.



	10	20	24	28	30	34	36	40
60	0.003141	0.003841	0.003143	0.004112	0.003331	0.004248	0.003556	0.004095
80	0.002341	0.004863	0.004053	0.004868	0.004861	0.004293	0.003143	0.004155
100	0.002894	0.004194	0.003719	0.004438	0.004329	0.004641	0.004883	0.004705
120	0.003426	0.004462	0.004709	0.004803	0.002633	0.004825	0.003806	0.003706
130	0.004128	0.004839	0.004598	0.003039	0.004563	0.004684	0.004598	0.004914
150	0.003814	0.004912	0.004728	0.004455	0.003813	0.004457	0.004193	0.003919
180	0.001214	0.004811	0.004883	0.004703	0.004351	0.004874	0.004596	0.003974
200	0.001341	0.004687	0.004426	0.004559	0.004911	0.004453	0.004796	0.004454

Gráfica 5.5. Valores de *C* obtenidos por el algoritmo.

La gráfica 5.6, muestra los valores de K obtenidos por el algoritmo para el caso 3



	10	20	24	28	30	34	36	40
60	-0.8343	-1.6243	-1.3483	-2.4528	-2.8314	-0.9314	-2.3031	-1.8857
80	-0.3423	-2.4354	-1.9565	-2.1864	-1.9796	-3.0193	-2.4348	-1.3427
100	-1.2843	-1.5913	-2.7907	-2.0531	-1.8251	-2.1453	-1.8053	-2.1273
120	-1.3461	-2.0136	-2.0932	-1.9002	-2.1893	-2.1169	-1.6396	-2.4047
130	-2.3428	-2.0136	-1.3372	-0.7175	-2.3141	-2.0178	-1.9646	-1.6836
150	-1.0431	-2.0342	-2.1128	-1.4099	-1.6735	-2.0199	-2.1362	-2.1105
180	-1.3834	-2.0449	-1.9782	-1.9915	-2.1505	-1.8356	-2.3017	-1.7509
200	-1.5667	-1.9276	-1.4954	-1.9256	-2.1393	-1.7737	-1.8763	-2.6865

Gráfica 5.6. Valores de K obtenidos por el algoritmo.

En esta gráfica se pueden observar valores para K muy cercanos al óptimo, al igual que en los casos anteriores.

Como se mencionó en el capítulo 4, los valores permitidos para C están dentro del intervalo [0, 0.005] y los de K dentro del intervalo [-4, 4].

En términos generales, y tomando en cuenta que el algoritmo parte de cero para encontrar los valores de las variables C y K, se pueden observar muy buenos resultados, los pares obtenidos (C y K) son muy cercanos a los óptimos.

La elección de los parámetros bajo los cuales el algoritmo trabaja de manera óptima es difícil establecerlos, debido a que los resultados de las pruebas hechas con anterioridad reflejan resultados muy similares, sin embargo se proponen en base a estos resultados los siguientes parámetros, los cuales arrojaron resultados satisfactorios.

Tamaño de la población 120.
Longitud del cromosoma 32.
Número de Generaciones 20.

Una vez obtenidos estos parámetros se realizaron dos pruebas más con diferentes archivos de datos con la finalidad de comprobar que dichos parámetros arrojan buenos resultados.

5.2 Prueba Número 2

Para la prueba número dos, se ocupó el archivo "radio50.dat", cuyos parámetros son; K = -1.00 y C = 0.005, se realizaron tres ejecuciones del algoritmo para dicha prueba.

El archivo "radio50.dat" se muestra en el anexo 3. Se recuerda que los parámetros bajo los cuales trabaja el algoritmo son

Tamaño de la Población: 120
Longitud del Cromosoma: 32
Número de Generaciones: 20

Los resultados obtenidos por el algoritmo fueron;

Valores Esperados		Valores Obtenidos	
C =	*K =*	*C =*	*K =*
0.0050	*-2.00*	0.00489	-0.9453
0.0050	*-2.00*	0.00475	-0.8711
0.0050	*-2.00*	0.00493	-0.9446

Tabla 5.1. Valores obtenidos por el algoritmo mediante el archivo "radio50.dat".

Se observa que los resultados son muy cercanos a los óptimos.

5.3 Prueba Número 3

Para la prueba número tres, se ocupó el archivo "rad50b.dat", cuyos parámetros son; $K = 0$ y $C = 0.005$, se realizaron tres ejecuciones del algoritmo para dicha prueba. Los parámetros que se ocupan para el algoritmo son los mismos que el la prueba 2. El archivo "rad50b.dat" se muestra en el anexo 4.

Los resultados obtenidos por el algoritmo fueron;

Valores Esperados		Valores Obtenidos	
C=	*K=*	*C=*	*K=*
0.0050	*0.00*	0.00489	0.03531
0.0050	*0.00*	0.00433	0.00493
0.0050	*0.00*	0.00484	0.00451

Tabla 5.2. Valores obtenidos por el algoritmo mediante el archivo "rad50b.dat".

Se observa que los resultados son muy cercanos a los óptimos.

5.4 Prueba Número 4

A partir de estos resultados, se realizó un programa que trabaja en dos etapas, la primera es la ya explicada hasta este momento, y en la segunda, se reducen los rangos de búsqueda para obtener todavía una mejor solución.

La primera parte del programa arroja un resultado de C y otro de K, a estos resultados se les hace la reducción antes mencionada, por ejemplo, si el resultado de $C = 0.00489$, se puede proponer como reducción 0.002, lo cual quiere decir que ahora el espacio de búsqueda de la solución quedará dentro del rango;

$$[0.00489 - 0.002 , 0.00489 + 0.002].$$

A continuación se presenta el diagrama de flujo correspondiente a dicho programa.

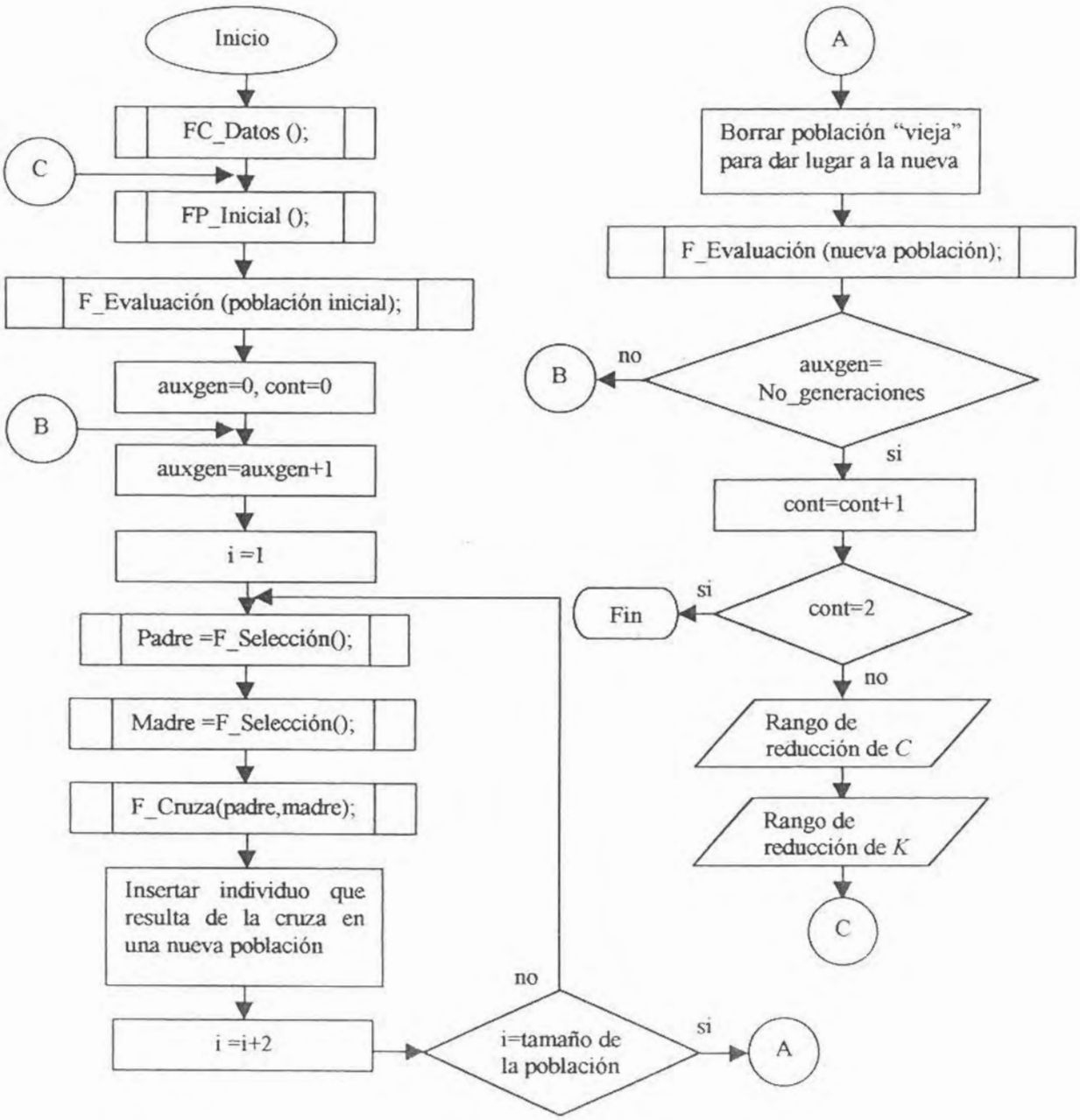


Figura 5.1. Diagrama de flujo del programa de dos etapas.

Como resultado de esta prueba se tiene que el algoritmo obtiene valores aún más cercanos a los óptimos ya que el rango de búsqueda se reduce. A continuación se presentan los resultados obtenidos.

Para esta prueba, se usó el archivo "radio50.dat", se realizaron tres corridas del programa, obteniendo los siguientes resultados.

Primera Etapa				Segunda Etapa	
Valores Esperados		Valores Obtenidos		Valores Finales	
C=	K=	C=	K=	C=	K=
0.0050	-1.00	0.00483	-1.0562	0.00492	-1.0056
0.0050	-1.00	0.00492	-1.2365	0.00499	-0.9948
0.0050	-1.00	0.00473	-0.8953	0.00494	-1.0048

Tabla 5.3. Resultados obtenidos por el algoritmo.

Como se puede observar en la tabla 5.3, los resultados finales están muy cerca del óptimo global.

5.5 Prueba Número 5

Para la prueba número 5, se tomaron datos de una superficie asférica, la cual se originó con un solo coeficiente de deformación A_1, quedando R_{iT} definida como;

$$R_{iT} = \frac{\left\{1+\left[\frac{CS}{\sqrt{1-(K+1)C^2S^2}}+6A_1S^5\right]^2\right\}^{3/2}}{\frac{C}{\left[1-(K+1)C^2S^2\right]^{3/2}}+30A_1S^4}, \qquad (5.1)$$

En este caso, el cromosoma se divide en tres partes, ya que existen tres variables (C, K y A_1), para esta prueba, se realizaron 2 etapas, en la primera se encuentran solamente los valores de C y K que mejor se ajustan a la superficie asférica, posteriormente se hace una reducción de rangos solo para C, el objetivo de esta primera parte, es la reducir el espacio de búsqueda para así obtener el óptimo, en la segunda etapa se utiliza la función de evaluación definida por la ecuación (5.1), utilizando el archivo "ras20.dat", el cual es presentado en el anexo 5, y que tiene los siguientes parámetros;

$C = 0.005, K = -1.00, A_1 = 0.0003.$

Se realizaron tres corridas del algoritmo obteniendo los siguientes resultados;

Primera Etapa

Valores Esperados			Valores Obtenidos	
C=	*K=*	*A1=*	*C=*	*K=*
0.005	*-1.00*	*0.0003*	*0,00283*	*-2.0062*
0.005	*-1.00*	*0.0003*	*0.00325*	*-1.5365*
0.005	*-1.00*	*0.0003*	*0.00235*	*-2.2802*

Segunda Etapa

Reducción	Valores Finales		
C=	*C=*	*K=*	*A1=*
0.0030	*0.00382*	*-1.9542*	*0.000851*
0.0025	*0.00452*	*-2.0251*	*0.000532*
0.0030	*0.00422*	*-1.7573*	*0.000482*

Tabla 5.4. Resultados obtenidos por el algoritmo.

Como se puede observar los valores arrojados por el algoritmo no son tan aceptables como los que se vieron en las pruebas pasadas, esto es por que el coeficiente de deformación "afecta" de manera significativa los valores de C y K.

Las siguientes figuras muestran ejemplos de una superficie cónica (figura 5.2), una asférica (figura 5.3) y por último la comparación de una cónica con una asférica con respecto a z (figura 5.4).

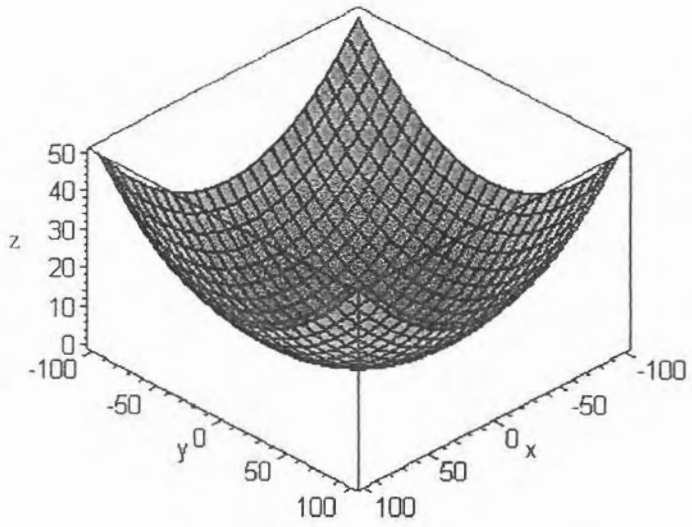


Figura 5.2. Superficie cónica $C = 0.005$ y $K = -1.00$.

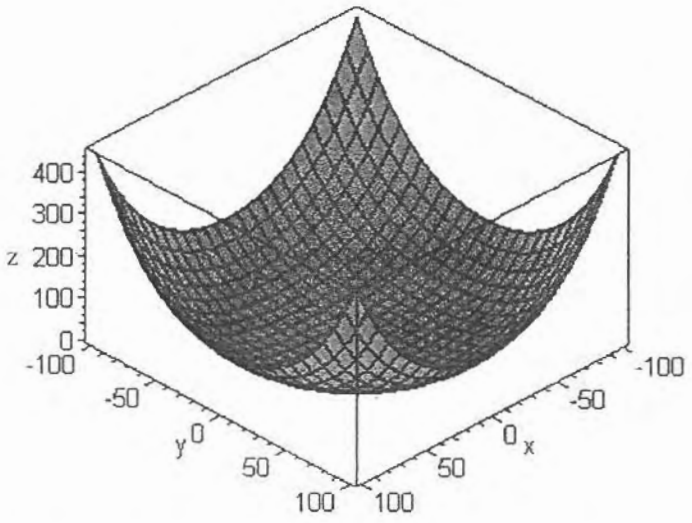


Figura 5.3. Superficie asférica $C = 0.005$, $K = -1.00$ y $A_1 = 0.0003$

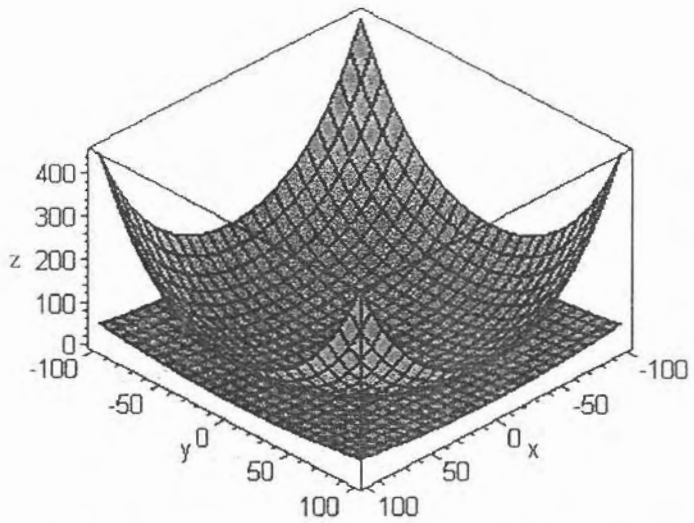


Figura 5.4. Comparación de las dos superficies anteriores figuras (5.2) y (5.3) con respecto a *z*

5.6 Conclusiones.

Los resultados obtenidos por el algoritmo para las primeras cuatro pruebas son muy buenos, cuando se realiza la prueba cuatro, es cuando se observan los mejores resultados, ya que estos tienen desviaciones del orden de 1.2 % con respecto al valor esperado, esto se debe a la reducción de rangos, ya que con esta acción, el espacio de búsqueda se reduce.

La prueba número cinco, es en la que se observan resultados poco cercanos a los óptimos, sin embargo, no son resultados muy alejados a estos, por lo que se propone que los valores arrojados por el algoritmo se utilicen como puntos de partida para utilizar un método de búsqueda local como el de mínimos cuadrados amortiguados.

Conclusiones

Se implementó un Algoritmo Genético el cual obtiene la forma de una superficie cónica convexa, a partir de radios de curvatura locales.

Los valores arrojados por el algoritmo son muy satisfactorios, sin embargo se encontraron mejores resultados al reducir el espacio de búsqueda de la solución (cuando el algoritmo trabaja en dos etapas).

Al almacenar los cromosomas y la información de estos (evaluación), en listas ligadas, se optimiza el uso de la memoria de la computadora, ya que las listas ligadas permiten la asignación dinámica de memoria, así como la liberación de la misma cuando la información de la lista ya no se utiliza.

Mediante las pruebas realizadas, se establecieron los parámetros (tamaño de la población, longitud del cromosoma y número de generaciones) bajo los cuales, y para el problema que se esta resolviendo en particular, el algoritmo obtiene soluciones cercanas al óptimo global (desviación de alrededor de 1.2 %).

Las soluciones encontradas para superficies asféricas arrojaron desviaciones con respecto al óptimo global de (alrededor de 4.2 %). Esto se debe al hecho de la no linealidad que presentan los coeficientes de deformación en la función de evaluación.

Por los que se propone como trabajo futuro, para superficies asféricas, implementar un algoritmo híbrido, el cual, utiliza la técnica de algoritmos genético para encontrar un punto inicial de partida y con un método de búsqueda local (i.e. mínimos cuadrados amortiguados), encontrar el óptimo global.

Anexos

Programa Principal

```
#include "base.h"
#include "boton.h"
#include "vents.h"
#include "3d.h"
# include <time.h>

# define h 200
# define mo 4
# define conver 1.0e15
#define tamx 100
#define tamy 100

// Prototipos de funciones

void FC_Datos();
void FP_Inicial();
void F_EvaluacionCo(struct Lista_pob *);
int F_Seleccion();
void F_Cruza(int,int);
void free_pob(struct Lista_pob *,int);
void free_val(struct Lista_val *,int);
void grafica(double valC,double valK);
double zg(double xg, double yg, double Cg, double Kg);

// Definicion de las estructuras

struct Lista_pob{
        unsigned char Cromo;
        struct Lista_pob *next_nodo;
        }*aux_ptrLp=NULL,*main_ptrLp,*cruza_ptrLp,
         *axpadre_ptrLp=NULL,*padre_ptrLp,*axmadre_ptrLp=NULL,*madre_ptrLp,
         *newax_ptrLp,*newmain_ptrLp,*del_ptrLp=NULL;

typedef struct Lista_pob NODO;
typedef NODO *Gen;

Gen New_Nodo()
{
        return((Gen) malloc(sizeof(NODO)));}


struct Lista_val{
        int No_Cromo;
        double valor_dec;
        double valor_dec1;
        double RT;
        double G;
        struct Lista_val *next_nodo;
        }*aux_ptrLv=NULL,*main_ptrLv,*aux_LvAs=NULL,
        *main_LvAs,*del_ptrLv;
```

```
typedef struct Lista_val NODO_1;
typedef NODO_1 *Gen_1;

Gen_1 New_nodo_1()
{
        return((Gen_1) malloc(sizeof(NODO_1)));
}

// Variables Globales

int Tpob, Lcromo,i,j,jj,jjj,Aux,No_generaciones,generacion,proce;
int TotalBits, NoBits, NoNodos,Val_max,Nodos_p,if_p;
unsigned char valor_ass,valor_ass1,mascara=1<<7;
double C,Laux,Km,maximo,con_10,Cs,acum,Ks,Lauxs,Ro,Gaux;
double A1,A1aux,G_aux,Ksas_aux,Csas_aux;
double Suma_acu;
int padre,madre,aux_pama,hlp=0;
time_t t;

void main()
{
        proce=15;
        generacion=0;
        FC_Datos();
        srand((unsigned)time(&t));
        FP_Inicial();
        F_EvaluacionCo(aux_ptrLp);
        ventana mera(0,0,799,599,"Superficies Asfericas");     //definición del objeto ventana,
                                                               //coordenadas y titulo
        boton out(650,4,786,"Salir","Terminar Programa");     //def. del botón, coordenadas.
        ig();                          //rutina de inicialización de gráficos 800x600 265 colores
        mera.open();                   //abre la ventana (la dibuja)
        mousehide();                   //esconde el mouse
        marco(3,25,797,574,on);        //dibuja lo "sumido" de la ventana
        mouseshow();                   //muestra el mouse
        out.dib();                    //dibuja el boton de salida
        write(4,300,30,"Superfice Optima");
        write(1,630,50,"Valores Optimos");
        write(1,650,92,"C =  %.4f",0.005);
        write(1,650,107,"K = %.4f",-2.00);
        coor_y=coor_y-200;
        grafica(0.005,-2.00);
        write(1,15,280,"Procesando ...");

do{
        write(5,proce,295,"Û");
        generacion=generacion+1;

        if(newax_ptrLp==NULL)
                {
                newax_ptrLp=New_Nodo();
                }

        newmain_ptrLp=newax_ptrLp;
        newmain_ptrLp->Cromo=0;
```

```
        aux_pama=1;
for(i=1;i<=Tpob;i=i+2)
{
        do{
          padre=F_Seleccion();    //Llamadas para seleccionar padres.
          madre=F_Seleccion();
          }
          while(padre==madre);
          F_Cruza(padre,madre); //Llamada para realizar la cruza de
                                //los padres.

          padre_ptrLp=axpadre_ptrLp;
          valor_ass=padre_ptrLp->Cromo;
          for(j=1;j<=Lcromo;++j,aux_pama++)
          {
                  if(valor_ass & mascara)
                  {
                  newmain_ptrLp->Cromo=(newmain_ptrLp->Cromo<<=1)|1;}
                  else
                  {
                  newmain_ptrLp->Cromo=(newmain_ptrLp->Cromo<<=1)|0;}
                  valor_ass<<=1;
                  if(j%8==0)
                  {
                  padre_ptrLp=padre_ptrLp->next_nodo;
                  valor_ass=padre_ptrLp->Cromo;
                  }
                  if(aux_pama%8==0)
                  {
                  newmain_ptrLp->next_nodo=New_Nodo();
                  newmain_ptrLp=newmain_ptrLp->next_nodo;
                  newmain_ptrLp->Cromo=0;
                  }
        }

/************** Liberando memoria para padre ************/

        Nodos_p=Lcromo/8;
        if_p=Lcromo%8;
        if(if_p!=0)
                Nodos_p++;

         padre_ptrLp=axpadre_ptrLp;
         free_pob(padre_ptrLp,Nodos_p);
         axpadre_ptrLp=NULL;

/*************** Obteniendo valores de la madre **********/
          madre_ptrLp=axmadre_ptrLp;
          valor_ass=madre_ptrLp->Cromo;
          for(j=1;j<=Lcromo;++j)
          {
                  if(valor_ass & mascara)
                  {
                  newmain_ptrLp->Cromo=(newmain_ptrLp->Cromo<<=1)|1;}
                  else
```

```
                {
                newmain_ptrLp->Cromo=(newmain_ptrLp->Cromo<<=1)|0;}
                valor_ass<<=1;
                if(j%8==0)
                {
                madre_ptrLp=madre_ptrLp->next_nodo;
                valor_ass=madre_ptrLp->Cromo;
                }
                if(aux_pama%8==0)
                {
                newmain_ptrLp->next_nodo=New_Nodo();
                newmain_ptrLp=newmain_ptrLp->next_nodo;
                newmain_ptrLp->Cromo=0;
                }
                aux_pama++;
        }

/************** Liberando memoria para madre ************/

        madre_ptrLp=axmadre_ptrLp;
        free_pob(madre_ptrLp,Nodos_p);
        axmadre_ptrLp=NULL;
}

        while(aux_pama%8!=0)
                {
                newmain_ptrLp->Cromo<<=1;
                aux_pama++;
                }
                newmain_ptrLp->Cromo<<=1;
                newmain_ptrLp->next_nodo=NULL;

/********Liberando memoria para la lista de valores********/

        main_ptrLv=aux_ptrLv;  //inicio de la lista
        free_val(main_ptrLv,Tpob);
        aux_ptrLv=NULL;

/******************Actualizando la lista ****************/

        main_ptrLp=aux_ptrLp;
        newmain_ptrLp=newax_ptrLp;
        for(j=1;j<NoNodos;++j)
                {
                main_ptrLp->Cromo=newmain_ptrLp->Cromo;
                main_ptrLp=main_ptrLp->next_nodo;
                newmain_ptrLp=newmain_ptrLp->next_nodo;
                }
                main_ptrLp->Cromo=newmain_ptrLp->Cromo;
```

```
/********Llamada para evaluar nueva poblacion*************/

        F_EvaluacionCo(aux_ptrLp);

/*********Liberando memoria de la nueva poblacion********/

        newmain_ptrLp=newax_ptrLp;
        free_pob(newmain_ptrLp,NoNodos);
        newax_ptrLp=NULL;

        proce=proce+8;
        }while(generacion<No_generaciones); // hasta que se cumpla
                                            // numero de generaciones
        write(2,15,280,"Procesando ...");
        write(1,15,280,"Finalizado");

        /*************** liberando memoria ***************/

        madre_ptrLp=axmadre_ptrLp;
        free_pob(madre_ptrLp,Nodos_p);
        axmadre_ptrLp=NULL;

        padre_ptrLp=axpadre_ptrLp;
        free_pob(padre_ptrLp,Nodos_p);
        axpadre_ptrLp=NULL;

        newmain_ptrLp=newax_ptrLp;
        free_pob(newmain_ptrLp,NoNodos);
        newax_ptrLp=NULL;


        write(4,300,300,"Superficie Obtenida");
        write(1,630,290,"Valores Obtenidos");
        write(1,630,303,"por el Algoritmo");
        write(1,650,345,"C =  %.4f",Cs);
        write(1,650,360,"K = %.4f",Ks);
        coor_y=coor_y+280;
        grafica(Cs,Ks);      //se dibuja la gráfica con los valores obtenidos por el AG.
do                           //ciclo del cual depende el programa, por el censado del mouse
{
 status();                 //checa el estado del mouse.
 out.que();                  //verifica que pasa con el botón de salida
 if(out.edo()==sel && hlp!=1) //si esta seleccionado y su ayuda no esta mostrada...
 {
        hlp=1;
        mera.msg(out.help()); //muestra en la ventana, abajo, la propiedad de ayuda del botón
 }
 if(out.edo()==nprensel && hlp!=0)
{
boton2.edo()==nprensel
        hlp=0;                      //se resetea
        mera.nomsg();               //borra el área donde se escribe la ayuda
 }
 }while(out.edo()!=pre);     //y esto se repite hasta que se presione el botón de salida
 out.pop();                 //botón de salida que se presiono.
```

```
nog();

        main_ptrLv=aux_ptrLv;  //inicio de la lista
        free_val(main_ptrLv,Tpob);
        aux_ptrLv=NULL;


        // Liberando memoria

        main_ptrLp=aux_ptrLp;
        free_pob(main_ptrLp,NoNodos);
        aux_ptrLp=NULL;

}// end main

// Función de la gráfica

void grafica(double valC,double valK)
{
 double cte=1;                  //separación de los puntos (vértices)
 double alt=2;                  //altura
 double intervalo=0.5;          //cada cuanto se incrementa la diferencia de los valores para cada vértice
 double Zg, max=0, min=0, pro=1;
 mousehide();
 setview(5,27,795,572);   //define el área de lo sumido de la ventana para que el dibujo no se pase de ahí
 for(double xg=0; xg<=(tamx-intervalo); xg+=intervalo)
 {
  for(double yg=0; yg<=(tamy-intervalo); yg+=intervalo)
  {
        Zg=zg(xg,yg,valC,valK);
        if(Zg>=max)
        {
         max=Zg;
        }
        if(Zg<=min)
        {
         min=Zg;
        }
  }
 }
 pro=(240)/(max-min);
 for(xg=-(tamx-intervalo); xg<=(tamx-intervalo); xg+=intervalo)//barrido de valores en x
 for(double yg=-(tamy-intervalo); yg<=(tamy-intervalo); yg+=intervalo)//barrido de valores en y
 {
  Zg=zg(xg,yg,valC,valK);
  punto(1,-6-floor(pro*Zg),xg*cte,yg*cte,-alt*Zg);
  punto(1,-6-floor(pro*Zg),xg*cte,yg*cte,-alt*Zg);
 }
 setview(0,0,799,599);      //define el área de dibujo como se da por default.
 mouseshow();
}
```

```
//Retorno de Z

double zg(double xg, double yg, double Cg, double Kg)
{

 return (Cg*(pow(xg,2.00)+pow(yg,2.00)))/(1+sqrt(1-(Kg+1)*pow(Cg,2.00)*(pow(xg,2.00)+pow(yg,2.00))));
}

// Función de evaluación

void F_EvaluacionCo(struct Lista_pob *main_ptrLp)
{

C=0.0;Km=0.0;acum=0.0;
valor_ass = main_ptrLp->Cromo;
int log_cromo=1,dox,temp=0,X,Y;
double conv_dec;
double roo,rt_aux,rt2,Re_aux;
float Re;
roo=0.0;rt_aux=0.0;rt2=0.0;Re_aux=0.0;G_aux=0.0;
Suma_acu=0.0;
FILE *archi;
dox=Lcromo/2;
if(aux_ptrLv==NULL)
        {
        aux_ptrLv = New_nodo_1();
        }
main_ptrLv = aux_ptrLv;

for(i=1;i<=Tpob;++i)
{
        main_ptrLv->No_Cromo=i;
        main_ptrLv->valor_dec=0.0;
        main_ptrLv->valor_dec1=0.0;
        main_ptrLv->RT=0.0;
        main_ptrLv->G=0.0;
        conv_dec=1;
        temp=0;
        for(j=1;j<=Lcromo;j++)
                {               //conversión a decimal del cromo
                if(valor_ass & mascara)
                {
                main_ptrLv->valor_dec = main_ptrLv->valor_dec + conv_dec;
                }
                conv_dec = conv_dec * 2;
                valor_ass = valor_ass<<=1;
                temp++;
                                //valor_dec = C; valor_dec1 = K
                if(dox==temp)
                {
                main_ptrLv->valor_dec1 = main_ptrLv->valor_dec;
                main_ptrLv->valor_dec = 0;
                conv_dec = 1;
                }
```

```
                if(log_cromo%8==0)
                        {
                        main_ptrLp = main_ptrLp->next_nodo;
                        valor_ass = main_ptrLp->Cromo;
                        log_cromo = 0;
                        }
                log_cromo++;
        }//end for

        main_ptrLv->next_nodo = New_nodo_1();
        main_ptrLv=main_ptrLv->next_nodo;
}  //end for

    for(i=1;i<=Tpob;i++)
        {           //Mapeo para encontrar de C

                C=main_ptrLv->valor_dec/((pow(2,dox)-1)*h);
                        // Mapeo para encontrar K
                Laux=main_ptrLv->valor_dec1*8/(pow(2,dox)-1);
                Km=mo-Laux;

                //Obteniendo los valore de x, y, Rie del archivo rad20.dat

                if ((archi=fopen("a:\\rad20.dat", "r"))==NULL)
                printf("No se puede encontrar el archivo\n");
                else{
                    main_ptrLv->RT=0.0;
                    main_ptrLv->G=0.0;
                    G_aux=0.0;
                    fscanf(archi,"%d%d%f",&X,&Y,&Re);

                //sumatoria de RT para todos los valores Ro
                //y para todos los cromosomas.
                        while(!feof(archi))
                        {
                        roo=pow(X,2)+pow(Y,2);
                        rt_aux=1-(Km*pow(C,2.0000)*roo);
                        if(rt_aux<0)
                                {
                                fscanf(archi,"%d%d%f",&X,&Y,&Re);
                                }
                        else
                                {
                                rt2=sqrt(rt_aux);
                                rt2=pow(rt2,3.0000);
                                main_ptrLv->RT=1/C*rt2;
                                Re_aux=Re-main_ptrLv->RT;
                                main_ptrLv->G=pow(Re_aux,2.0000);
                                main_ptrLv->G=main_ptrLv->G+G_aux;
                                G_aux=main_ptrLv->G;
                                fscanf(archi,"%d%d%f",&X,&Y,&Re);
                                }
                        }
                fclose(archi);
                  }
```

```
                acum=conver-main_ptrLv->G;
                Suma_acu=acum+Suma_acu; //Suma acumulativa
                main_ptrLv = main_ptrLv->next_nodo;
        }//end for

                // Obteniendo los valores máximo de G(i)

main_ptrLv=aux_ptrLv;   //Retorno al principio de la lista
maximo=conver-main_ptrLv->G;
Val_max=1;
        for(i=2;i<=Tpob;i++)
        {
         main_ptrLv = main_ptrLv->next_nodo;
         con_10=conver-main_ptrLv->G;
         if(con_10>maximo)       // Buscando el mayor
                {
                maximo=con_10;
                Val_max=i;
                }
        }
                main_ptrLv=aux_ptrLv;
                i=1;
                while(i<Val_max)
                {
                main_ptrLv=main_ptrLv->next_nodo;
                i++;
                }
                Cs=main_ptrLv->valor_dec/((pow(2,dox)-1)*h);

                        // Mapeo para encontrar K
                Lauxs=main_ptrLv->valor_dec1*8/(pow(2,dox)-1);
                Ks=mo-Lauxs;
                Gaux=main_ptrLv->G;
}

//Función para generar pob inicial

void FP_Inicial()
{
        TotalBits=Tpob*Lcromo;
        NoNodos=TotalBits/8;
        NoBits=TotalBits%8;

        if(NoBits!=0)
                NoNodos++;

if(aux_ptrLp==NULL)
                {
                aux_ptrLp=New_Nodo();
                }
main_ptrLp = aux_ptrLp;

for(i = 1; i < NoNodos; i++)
        {
        main_ptrLp->Cromo = rand() % 256;
```

```
        main_ptrLp->next_nodo = New_Nodo();
        main_ptrLp = main_ptrLp->next_nodo;
        }
main_ptrLp->Cromo=rand()%256;
main_ptrLp->next_nodo=NULL;
main_ptrLp = aux_ptrLp;
}

//Función de Selección

int F_Seleccion()
{
double sumpar=0.0;
int i_cromo=0;
double pastel=0.0;
do{
pastel=rand()%1000;
 }while(pastel==0);
pastel=pastel/1000;
pastel=pastel*Suma_acu;
main_ptrLv=aux_ptrLv;

do{
        i_cromo++;
        sumpar=(conver-main_ptrLv->G)+sumpar;
        main_ptrLv = main_ptrLv->next_nodo;
  }
  while(sumpar<pastel && i_cromo<Tpob);
  return (i_cromo);
}

//Función de Cruza

void F_Cruza(int novio, int novia)
{
int Punto_cruza=1 + (rand() % (Lcromo - 1));

/**** Localizando en donde se encuentra el "padre" dentro de la lista ************
                guardándolo en main_ptrLp*******************************/

main_ptrLp=aux_ptrLp;
valor_ass=main_ptrLp->Cromo;
j=1;
while(j<=((novio-1)*Lcromo))
        {
        valor_ass<<=1;
        if(j%8==0)
                {
                main_ptrLp=main_ptrLp->next_nodo;
                valor_ass=main_ptrLp->Cromo;
                }
        j++;
        }
```

```
/*********Localizando hasta donde se encuentra la "madre" dentro de la lista********
        el valor se almacena en cruza_ptrLp*********************************/

cruza_ptrLp=aux_ptrLp;
valor_ass1=cruza_ptrLp->Cromo;
jj=1;
while(jj<=((novia-1)*Lcromo))
        {
        valor_ass1<<=1;
        if(jj%8==0)
                {
                cruza_ptrLp=cruza_ptrLp->next_nodo;
                valor_ass1=cruza_ptrLp->Cromo;
                }
        jj++;
        }


/******* Se posiciona en que punto se realizará la cruza
        en el cromosoma padre y el cromosoma madre *******/

//Inicializando lista para almacenar padre
if(axpadre_ptrLp==NULL)
                {
                axpadre_ptrLp=New_Nodo();
                }
padre_ptrLp=axpadre_ptrLp;
padre_ptrLp->Cromo=0;

//Inicializando lista para almacenar madre
if(axmadre_ptrLp==NULL)
                {
                axmadre_ptrLp=New_Nodo();
                }
madre_ptrLp=axmadre_ptrLp;
madre_ptrLp->Cromo=0;

jjj=1;
while(jjj<=Punto_cruza)
        {       // cromosoma padre

        if(valor_ass & mascara)
                {padre_ptrLp->Cromo=(padre_ptrLp->Cromo<<=1)|1;}
        else
                {padre_ptrLp->Cromo=(padre_ptrLp->Cromo<<=1)|0;}

                  // cromosoma madre

        if(valor_ass1 & mascara)
                {madre_ptrLp->Cromo=(madre_ptrLp->Cromo<<=1)|1;}
        else
                {madre_ptrLp->Cromo=(madre_ptrLp->Cromo<<=1)|0;}

        valor_ass1<<=1;
        valor_ass<<=1;
```

```
        if(jj%8==0)
                {
                cruza_ptrLp=cruza_ptrLp->next_nodo;
                valor_ass1=cruza_ptrLp->Cromo;}
        if(j%8==0)
                {
                main_ptrLp=main_ptrLp->next_nodo;
                valor_ass=main_ptrLp->Cromo;}
        if(jjj%8==0)
                {
                padre_ptrLp->next_nodo=New_Nodo();
                padre_ptrLp=padre_ptrLp->next_nodo;
                padre_ptrLp->Cromo=0;
                madre_ptrLp->next_nodo=New_Nodo();
                madre_ptrLp=madre_ptrLp->next_nodo;
                madre_ptrLp->Cromo=0;}
        j++;
        jj++;
        jjj++;
}

/************* Se realiza la mezcla de los cromosomas progenitores **********
                dependiendo del punto de cruza dado */

while(jjj<=Lcromo)
        {
        if(valor_ass & mascara)
                {madre_ptrLp->Cromo=(madre_ptrLp->Cromo<<=1)|1;}
        else
                {madre_ptrLp->Cromo=(madre_ptrLp->Cromo<<=1)|0;}

        if(valor_ass1 & mascara)
                {padre_ptrLp->Cromo=(padre_ptrLp->Cromo<<=1)|1;}
        else
                {padre_ptrLp->Cromo=(padre_ptrLp->Cromo<<=1)|0;}
        valor_ass<<=1;
        valor_ass1<<=1;

        if(j%8==0)
                {
                main_ptrLp=main_ptrLp->next_nodo;
                valor_ass=main_ptrLp->Cromo;}
        if(jj%8==0)
                {
                cruza_ptrLp=cruza_ptrLp->next_nodo;
                valor_ass1=cruza_ptrLp->Cromo;}
        if(jjj%8==0)
                {
                padre_ptrLp->next_nodo=New_Nodo();
                padre_ptrLp=padre_ptrLp->next_nodo;
                padre_ptrLp->Cromo=0;

                madre_ptrLp->next_nodo=New_Nodo();
                madre_ptrLp=madre_ptrLp->next_nodo;
```

```
                madre_ptrLp->Cromo=0,}
        ii++;
        j++;
        iii++;
}

while(iii%8!=0)
        {
        padre_ptrLp->Cromo<<=1;
        madre_ptrLp->Cromo<<=1;
        iii++;
        }

padre_ptrLp->Cromo<<=1;
madre_ptrLp->Cromo<<=1;
padre_ptrLp->next_nodo=NULL;
madre_ptrLp->next_nodo=NULL;
}

// Función para liberar memoria de la lista de valores

void free_val(struct Lista_val *free_ptr,int Frp)
{
int nac;
for(nac=1;nac<Frp;nac++)
        {
        del_ptrLv=free_ptr;
        free_ptr=free_ptr->next_nodo;
        free(del_ptrLv);
        }
        del_ptrLv=free_ptr;
        free(del_ptrLv);
}

// Función para liberar memoria de la lista de la población

void free_pob(struct Lista_pob *free_ptr,int Frp)
{
int nac;
for(nac=1;nac<Frp;nac++)
        {
        del_ptrLp=free_ptr;
        free_ptr=free_ptr->next_nodo;
        free(del_ptrLp);
        }
        del_ptrLp=free_ptr;
        free(del_ptrLp);
}
```

```
//Funcion captura datos iniciales

void FC_Datos()

{
        Tpob=120;
        Lcromo=32;
        No_generaciones=20;
}
```

Librería "base.h"

```
#include "stdlib.h"
#include "string.h"
#include "dos.h"
#include "stdarg.h"
#include "math.h"
#include "conio.h"
#include "stdio.h"
#include "svgacc.h"

#define cls fillpage(0)
#define big setview(0,0,799,599)
enum Teclas{arr=72, aba=80, izq=75, der=77, Esc=27, enter=13, tabu=9, tab_shift=15, F1=59};
enum Bool{on=1, off=0, pre=2, sel, nprensel};
void ig();//inicio de graficos
void write(int x, int y, char *fmt, ...);
int coor_x=350;//coordenadas del origen
int coor_y=300;
PaletteData P;
int xr,yr,zr;
void rgb(int colnum, int a, int b, int c);
void status();
int huge DetectVGA256()
{
 return 3;
}

MouseCursor Arrow={
0,0,
  2, 4,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  2, 0, 4, 4,255,255,255,255,255,255,255,255,255,255,255,255,
255, 2, 0, 0, 4, 4,255,255,255,255,255,255,255,255,255,255,
255, 2, 0, 0, 0, 0, 4, 4,255,255,255,255,255,255,255,255,
255,255, 2, 0, 0, 0, 0, 0, 4, 4,255,255,255,255,255,255,
255,255, 2, 0, 0, 0, 0, 0, 0, 0, 4, 4,255,255,255,255,
255,255,255, 2, 0, 0, 0, 0, 0, 0, 0, 0, 4, 4,255,255,
255,255,255, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 4,
255,255,255,255, 2, 0, 0, 0, 4, 2, 2, 2, 2, 2, 2, 2,
255,255,255,255, 2, 0, 0, 0, 4,255,255,255,255,255,255,255,
```

```
255,255,255,255,255, 2, 0, 0, 4,255,255,255,255,255,255,255,
255,255,255,255,255, 2, 0, 0, 4,255,255,255,255,255,255,255,
255,255,255,255,255,255, 2, 0, 4,255,255,255,255,255,255,255,
255,255,255,255,255,255, 2, 0, 4,255,255,255,255,255,255,255,
255,255,255,255,255,255,255, 2, 4,255,255,255,255,255,255,255,
255,255,255,255,255,255,255, 2, 4,255,255,255,255,255,255,255,
255,255,255,255,255,255,255,255, 2,255,255,255,255,255,255,255,
255,255,255,255,255,255,255,255, 2,255,255,255,255,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
};

void ig(void)
{
 if(!whichvga())
 {
  printf("La tarjeta gráfica no tiene capacidad para esta resolución\n");
  exit(1);
 }
 if(whichmem() < 512)
 {
  printf("La memoria de la tarjeta gr fica no tiene capacidad para esta resolución\n");
  exit(1);
 }
 if(!whichmouse())
 {
  printf("No se detectó mouse, imposible ejecutar este programa\n");
  exit(1);
 }
 res800();
 fillpage(5);
 mouseenter();
 mouselocset(300,200);
 mousecursorset(&Arrow);
 mouseshow();
 rgb(1,255,255,255);
 rgb(2,96,96,175);
 rgb(3,176,176,208);
 rgb(4,230,240,5);
 rgb(5,220,10,30);
 for(int h=0;h<249;h++)
 {
  rgb(6+h,6+h,6+h,6+h);
 } palset(P,0,255);//se utiliza para actualizar la paleta nom.pal. col.ini col.fin
}
```

```
void nog(void)
{
 mousehide();
 mouseexit();
 restext();
 exit(0);
}

//Función para modificar la paleta de colores
void rgb(int colnum, int a, int b, int c)
{
 if(a>=255)a=255;
 if(b>=255)b=255;
 if(c>=255)c=255;
 P[colnum].r=floor(0.2470*a);
 P[colnum].g=floor(0.2470*b);
 P[colnum].b=floor(0.2470*c);
}

// Función para escribir
void write(int col, int x, int y, char *fmt, ...)
{
 va_list punt;
 char buff[80];
 va_start(punt, fmt);
 vsprintf(buff, fmt, punt);
 va_end(argptr);
 drwstring(AND,0,255,buff,x,y);
 drwstring(OR,col,0,buff,x,y);
}
void status()
{
 mousestatus(&xr,&yr,&zr);
}

// Función para dibujar el marco
void marco(int x1, int y1, int x2, int y2, int onoff)
{
 drwline(SET,(onoff==off)?3:0,x1,y1,x2-1,y1);
 drwline(SET,(onoff==off)?3:0,x1,y2-1,x1,y1);
 drwline(SET,(onoff==off)?0:3,x1,y2,x2,y2);
 drwline(SET,(onoff==off)?0:3,x2,y1,x2,y2);
 if(onoff==off)
 {
  drwline(SET,4,x1+1,y2-1,x2-1,y2-1);
  drwline(SET,4,x2-1,y1+1,x2-1,y2-1);
 }
 else if(onoff==on)
 {
  drwline(SET,4,x1+1,y1+1,x2-1,y1+1);
  drwline(SET,4,x1+1,y1+1,x1+1,y2-1);
 }
}
```

Librería "3d.h"

```
void punto(int opc, int col, float x, float y, float z);
void linea(int opc, int col, float x1, float y1, float z1, float x2, float y2, float z2);
void m_o(int mas_x, int mas_y);
float k=1;//constante que varia algo asi como la proporcion de altura de un grafico en 3D

//drwline(SET|AND|OR|XOR , PALETA[x] , x1,y1,x2,y2)

void punto(int opc, int col, float x, float y, float z)//transformacion 3D-2D
{
 drwpoint(opc,col,coor_x+(((-0.8660)*x)+(0.8660*y)),coor_y-(((-0.5)*x)+((-0.5)*y)+z)*k);
}

void linea(int opc, int col, float x1, float y1, float z1, float x2, float y2, float z2)//transformacion 3D-2D
{
 drwline(opc,col,coor_x+((-0.8660*x1)+(0.8660*y1)),coor_y-(((-0.5)*x1)+((-0.5)*y1)+z1)*k,coor_x+(((-
0.8660)*x2)+((0.8660)*y2)),coor_y-(((-0.5)*x2)+((-0.5)*y2)+z2)*k);
}

void m_o(int mas_x, int mas_y)//declaracion del origen (2D)
{
 coor_x+=mas_x; coor_y+=mas_y;
}
```

Librería "boton.h"

```
class boton//definicion del objeto boton
{
 public:
 boton(int=0,int=0,int=0,char *,char *);
 ~boton();
 void dib();//que se dibuje
 void que();//que pasa con el
 int edo();//retorna el estado en el que se encuentra
 void pop();//lo levanta despues de presionado, y resetea su edo.
 char *help();//retorna cadena de ayuda
 protected:
 int x1,y1,x2,y2,modo;//coordenadas y estado
 char *nombre;//cadena para lo que dice el boton
 char *hlp;//cadena de ayuda
};

boton::boton(int X1, int Y1, int X2, char *nom, char *help)
{
 x1=X1;
 y1=Y1;
 x2=X2;
 y2=y1+16;
 nombre=nom;
```

```
 modo=nprensel;//por default ni presionado ni seleccionado
 hlp=help;
}

boton::~boton()
{}

void boton::dib()
{
 mousehide();
 marco(x1,y1,x2,y2,off);
 drwfillbox(SET,2,x1+1,y1+1,x2-2,y2-2);
 write(2,((x2-x1)/2)+x1-(strlen(nombre)*4)+1,y1+2,"%s",nombre);
 write(0,((x2-x1)/2)+x1-(strlen(nombre)*4),y1+1,"%s",nombre);
 mouseshow();
}

void boton::que()
{
 if(xr>=x1 && xr<=x2 && yr>=y1 && yr<=y2)//si el mouse (xr,yr) esta dentro de sus
limites...
 {
  if(zr==0 && modo==nprensel)//si no se da click y esta en modo ni presionado ni
seleccionado...
  {
        modo=sel;//se pone como seleccionado
        mousehide();
        write(1,((x2-x1)/2)+x1-(strlen(nombre)*4),y1+1,"%s",nombre);//se resalta su letrero
        mouseshow();
  }
  if(zr==1 && modo==sel)//si se dio click y estaba seleccionado...
  {
        mousehide();
        marco(x1,y1,x2,y2,on);
        drwline(SET,4,x1+1,y1+1,x2-1,y1+1);
        drwline(SET,4,x1+1,y1+1,x1+1,y2-1);
        write(2,((x2-x1)/2)+x1-(strlen(nombre)*4),y1+1,"%s",nombre);
        write(1,((x2-x1)/2)+x1-(strlen(nombre)*4)+1,y1+2,"%s",nombre);
        mouseshow();
        delay(100);//retardo para que se vea cuando se sume (solo si fue muy rapido el
click)
        do
        {
         status();//checa el estado
        }while(zr!=0);//hasta que no se suelte el boton
        if(xr>=x1 && xr<=x2 && yr>=(y1) && yr<=(y1+15))//si quedo dentro del boton,
        {
         modo=pre;//se cambia a presionado
        }
        else
        {
         modo=nprensel;//de otra manera, se resetea su valor
         dib();// y se vuele a dibujar
        }
```

```
 }
}
else
if(modo==sel)//si no esta en sus limites, y estaba seleccionado
{
 modo=nprensel;//lo resetea
 dib();
}
}

int boton::edo()
{
 return modo;//regresa el estado en el que esta
}

void boton::pop()
{
 modo=nprensel;
 boton::dib();
}

char *boton::help()
{
 return (hlp);
}
```

Librería "Vents.h"

```
// declaracion de la clase ventana

class ventana
{
 public:
  ventana(int,int,int,int,char *);
  ~ventana();
  void open();
  void close();
  int maxx();
  int maxy();
  void justonit();
  void notthis();
  void msg(char *);
  void nomsg();
  void clear();
 private:
  int x1, y1, x2, y2;
 protected:
  char far *nombre;
};
```

```
ventana::ventana(int X1, int Y1, int X2, int Y2, char *titulo)
{
 x1=X1;
 x2=X2;
 y1=Y1;
 y2=Y2;
 nombre=new char far[strlen(titulo)+1];
 sprintf(nombre,"%s",titulo);
}

ventana::~ventana()
{
 delete nombre;
}

int ventana::maxx()
{
 return (x2-x1);
}

int ventana::maxy()
{
 return (y2-y1-16);
}

void ventana::open()
{
 mousehide();
 marco(x1,y1,x2,y2,off);
 drwfillbox(SET,2,x1+1,y1+1,x2-1,y2-1);
 drwfillbox(SET,0,x1+3,y1+3,x2-3,y1+21);
 marco(x1+2,y2-19,x2-2,y2-2,on);
 nomsg();
 write(1,x1+13,y1+5,"%s",nombre);
 mouseshow();
}

void ventana::close()
{
}

void ventana::justonit()
{
 setview(x1+3,y1+20,x2-3,y2-19);
}

void ventana::notthis()
{
 big;
}

void ventana::msg(char *ayuda)
{
 mousehide();
 drwfillbox(SET,2,x1+4,y2-17,x2-4,y2-4);
```

```
 write(1,x1+13,y2-17,"%s",ayuda);
 mouseshow();
}

void ventana::nomsg()
{
 mousehide();
 drwfillbox(SET,2,x1+4,y2-17,x2-4,y2-4);
 mouseshow();
}

void ventana::clear()
{
 mousehide();
 drwfillbox(SET,2,x1+3,y1+20,x2-3,y2-19);
 mouseshow();
}
```

ANEXO 2

Archivo " Rad20.Dat"

X	Y	R_{ie}
-100	-100	565.6854
- 90	- 90	487.0212
- 80	- 80	420.045
- 70	- 70	363.7554
- 60	- 60	317.2038
- 50	- 50	279.5085
- 40	- 40	249.8716
- 30	- 30	227.5987
- 20	- 20	212.1192
- 10	- 10	203.0075
0	0	200
10	10	203.0075
20	20	212.1192
30	30	227.5987
40	40	249.8716
50	50	279.5085
60	60	317.2038
70	70	363.7554
80	80	420.045
90	90	487.0212

ANEXO 3

Archivo "Radio50.dat"

X	Y	R_{ie}
-100	-100	367.4235
-96	-96	353.1150
-92	-92	339.5697
-88	-88	326.7673
-84	-84	314.6881
-80	-80	303.3129
-76	-76	292.6229
-72	-72	282.5999
-68	-68	273.2266
-64	-64	264.4858
-60	-60	256.3616
-56	-56	248.8384
-52	-52	241.9015
-48	-48	235.537
-44	-44	229.7318
-40	-40	224.4738
-36	-36	219.7516
-32	-32	215.555
-28	-28	211.8745
-24	-24	208.7019
-20	-20	206.0299
-16	-16	203.8523
-12	-12	202.1639
-8	-8	200.9608
-4	-4	200.2401
0	0	200
4	4	200.2401
8	8	200.9608
12	12	202.1639
16	16	203.8523
20	20	206.0299
24	24	208.7019
28	28	211.8745
32	32	215.555
36	36	219.7516
40	40	224.4738
44	44	229.7318
48	48	235.537
52	52	241.9015
56	56	248.8384
60	60	256.3616
64	64	264.4858
68	68	273.2266
72	72	282.5999
76	76	292.6229
80	80	303.3129
84	84	314.6881
88	88	326.7673
92	92	339.5697
96	96	353.1150

ANEXO 4

Archivo "Rad50b.dat"

X	Y	R_{ie}	X	Y	R_{ie}
-100	-100	200	4	4	200
- 96	- 96	200	8	8	200
- 92	- 92	200	12	12	200
- 88	- 88	200	16	16	200
- 84	- 84	200	20	20	200
- 80	- 80	200	24	24	200
- 76	- 76	200	28	28	200
- 72	- 72	200	32	32	200
- 68	- 68	200	36	36	200
- 64	- 64	200	40	40	200
- 60	- 60	200	44	44	200
- 56	- 56	200	48	48	200
- 52	- 52	200	52	52	200
- 48	- 48	200	56	56	200
- 44	- 44	200	60	60	200
- 40	- 40	200	64	64	200
- 36	- 36	200	68	68	200
- 32	- 32	200	72	72	200
- 28	- 28	200	76	76	200
- 24	- 24	200	80	80	200
- 20	- 20	200	84	84	200
- 16	- 16	200	88	88	200
- 12	- 12	200	92	92	200
- 8	- 8	200	96	96	200
- 4	- 4	200			
0	0	200			

ANEXO 5

Archivo "Ras20.dat"

X	Y	R_{ie}
-100	-100	5.433599 e +08
-90	-90	2.599212 e +08
-80	-80	1.139865 e +08
-70	-70	4.477394 e +07
-60	-60	1.522566 e +07
-50	-50	4252112
-40	-40	892867.3
-30	-30	119522.5
-20	-20	7062.281
-10	-10	64.68858
0	0	200
10	10	64.68858
20	20	7062.281
30	30	119522.5
40	40	892867.3
50	50	4252112
60	60	1.522566 e +07
70	70	4.477394 e +07
80	80	1.139865 e +08
90	90	2.599212 e +08

Referencias

1. T.H. Jamieson, *Optimization Techniques in Lens Desing*, Adam Hilger, London, 1971, Pags. 33-50.

2. C. Menchaca, Arquímedes Morales, D. Malacara, *Evaluación de Algunos Parámetros para Superficies Asféricas*, Centro de Investigaciones en Óptica, Reporte Técnico No. 3, León, Guanajuato México, Pags. 2-10.

3. Sergio Vázquez y Montiel, *Diseño de Sistemas Opticos usando Algoritmos Genéticos*, Departamento de Instrumentación Óptico-Digital, Tonantzintla Puebla, México, 1996, Pags. 4-43.

4. C.G. Wynne, *Lens Designing by Electronic Digital Computer, I*, Proc. Phys. Soc., 1959, Pags. 77-120.

5. R. Burden, J. Douglas, *Análisis Numérico*, Thomson Editores, 6° Edición, México, 1998.

6. Gunter Schulz, *Aspherics Surfaces*, Elsevier Science Publishers B.V., 1988, Pags. 351-357.

7. Raúl Cárdenas, *Generación de Curvas y Superficies*, TSIT, Universidad Politécnica de Madrid, Madrid España, 1989, Pags. 270-315.

8. C. Menchaca, Arquímedes Morales, D. Malacara, *Evaluación de Algunos Parámetros para Superficies Asféricas*, Centro de Investigaciones en Óptica, Reporte Técnico No. 3, León, Guanajuato México, Pags. 20-31.

9. Dennis G. Zill, *Cálculo con Geometría Analítica*, Grupo Editorial Iberoamérica, México, 1987, Pags. 698-709.

10. O. Cardona Núñez, A. Cornejo Rodríguez, *Significado de las Superficies Cuasticas en Óptica*, Revista Mexicana de Física 29 No. 2, México,1983, Pags. 247-253.

11. Nason Alvin, *Biología*, Limusa, México, 1985, Pags. 37-48, 665-685.

12. Oram, Hummer, Smoot, *Biology Living Systems*, Charles and Merril Pub. Co., USA, 1984, Pags. 159-200.

13. Carlos A. Coello Coello, *Soluciones Avanzadas*, Xview, S.A de C.V, México, 1995, Pags. 5-11.

14. Anselmo Pérez Serrada, *Una Introducción a la Computación Evolutiva*, Reporte Técnico, 1996, Pags. 1-15.

15. Lawrence Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991, Pags.1-20.

16. David E. Goldberg, *Genetic Algorithms*, Addison-Wesley Publishing Company, Inc., USA, 1989, Pags. 1-57.

17. Darrell Whitley, *A Genetic Algorithm Tutorial, Technical Report*, Department of Computer Science, Colorado State University, 1993.

18. Anselmo Pérez Serrada, *Una Introducción a la Computación Evolutiva*, Reporte Técnico, 1996, Pags. 30-32.

19. F. Herrera, *Algoritmos Genéticos: Fundamentos, Extensiones y Aplicaciones*, Reporte Técnico, Universidad de Granada, España, 1994.

20. Holland, J.H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975, Pags. 205-220.

21. H.M. Deitel, *Como Programar en C/C++*, Prentice Hall, 2° Ed., México, 1995.

22. Daniel.Malacara, *Optical Shop Testing*, John Wiley & Sons, Canadá, 1986, Pags. 479-483.

23. Javier Galve/Juan González, *Algoritmica: Diseño y Análisis de Algoritmos Funcionales e Imperativos*, Addison-Wesley Iberoamericana, México, 1993, Pags. 137-140, 418-423.

24. David A. Atchison, *On the Description of Zonal Aspheric Surfaces*, Amrican Journal of Optometry & Physiological Optics, USA, 1986, Pags. 14-15.