Capítulo 4. Implementación de algoritmos

Ya se han analizado las tres propuestas con las que se va a trabajar en lo que resta de este proyecto, el siguiente paso es diseñar un programa que pueda realizar la tarea requerida para cada modelo. Se propone utilizar el paquete MATLAB[1] para poder simular cada modelo debido a que este presenta herramientas que facilitan el cálculo de operaciones que resultan trascendentales para los modelos pero que pueden tomar mucho tiempo en realizarse, tal sea el caso del cálculo de gradientes, operaciones matriciales, interfaces gráficas, etc.

Los modelos que se manejan en este capítulo se basan en los algoritmos diseñados por Xu y Prince. Estos modelos se tomaron debido a que utilizan las ventajas que MATLAB tiene (operaciones matriciales, cajas de herramientas para procesado digital de imágenes), haciendo así un modelo fácil de programar y con buenos resultados. Cabe resaltar que estos algoritmos fueron solo utilizados para simulaciones y demostrar el funcionamiento de los modelos. Algunas definiciones se cambiaron respecto al programa original y también se cambió la interfase gráfica para ubicar los puntos de control, también se definen diferentes fuerzas externas dependiendo el ambiente en el que se opere.

4.1 Algoritmo para el modelo de Kass (modelo tradicional)

En este punto es pertinente hacer referencia al capítulo 1 para comprender el diseño del algoritmo del modelo de Kass o modelo tradicional. Este se muestra en la figura 4.1 en donde se señala que una vez que se descarga la imagen se convoluciona la imagen con una mascarilla correspondiente a una función Gaussiana (Apéndice A), lo cual genera una imagen desvanecida con la finalidad de hacer más amplio el borde después de obtener el gradiente para así obtener la fuerza externa. Por otro lado existe otro proceso correspondiente a la elección de los puntos de control que se hace de forma manual como se muestra en la figura 4.2, en donde el eje representa la posición donde se pondrán los puntos de control de inicio que se muestran como cuadros rojos.

Obtenido lo anterior es posible empezar a deformar el modelo, sin embargo este no se puede iterar muchas veces, ya que los puntos de control pueden unirse mucho o separase, lo que provoca que las ecuaciones no tengan sentido ya que cada punto de control depende de su punto anterior y posterior. Es por ello que se propone un proceso en donde se agregan o quiten puntos de control dependiendo la necesidad acorde a un umbral de distancias máximo y mínimo definido.

[1] Se utilizó la versión 6.2 y 7.1 de MATLAB (no se tuvieron diferencias en los resultados en ambas versiones). El programa se implementó en una computadora portátil GATEWAY ® modelo MX6618m con procesador Intel ® Celeron ® y Microsoft Windows ® XP.

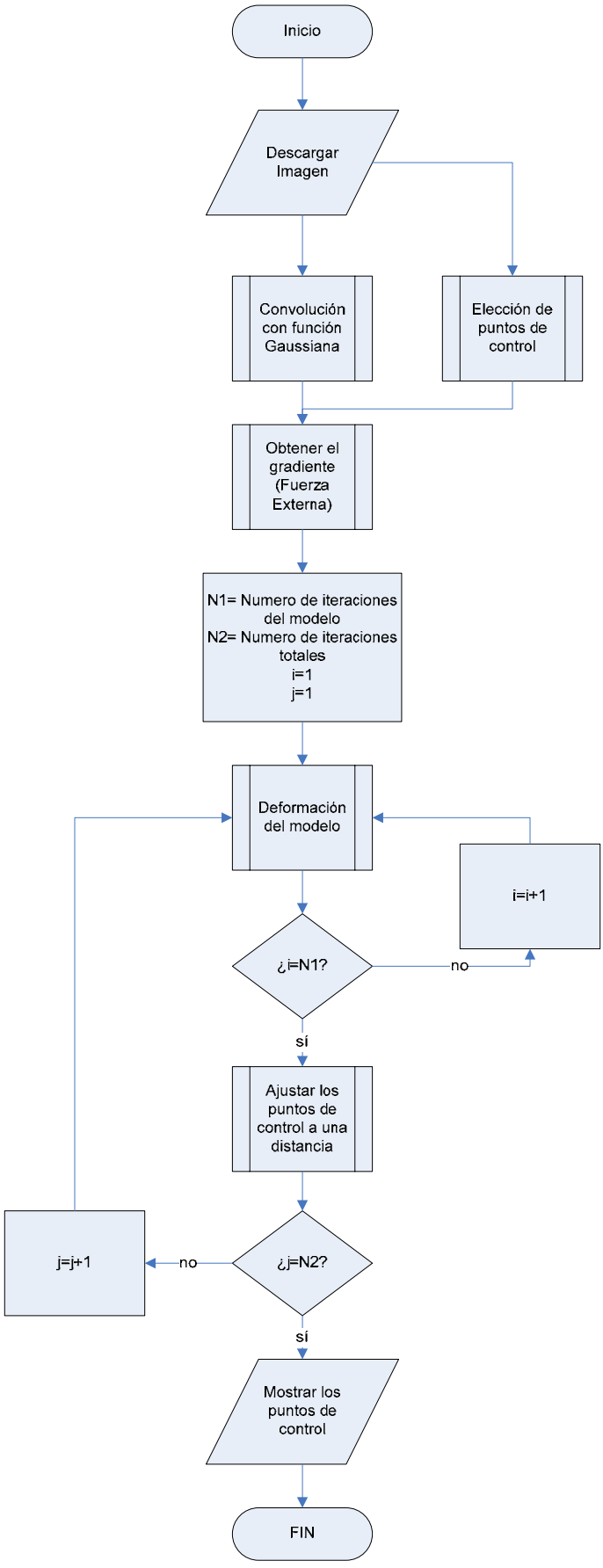

Figura 4.1 *Algoritmo para el modelo de Kass*

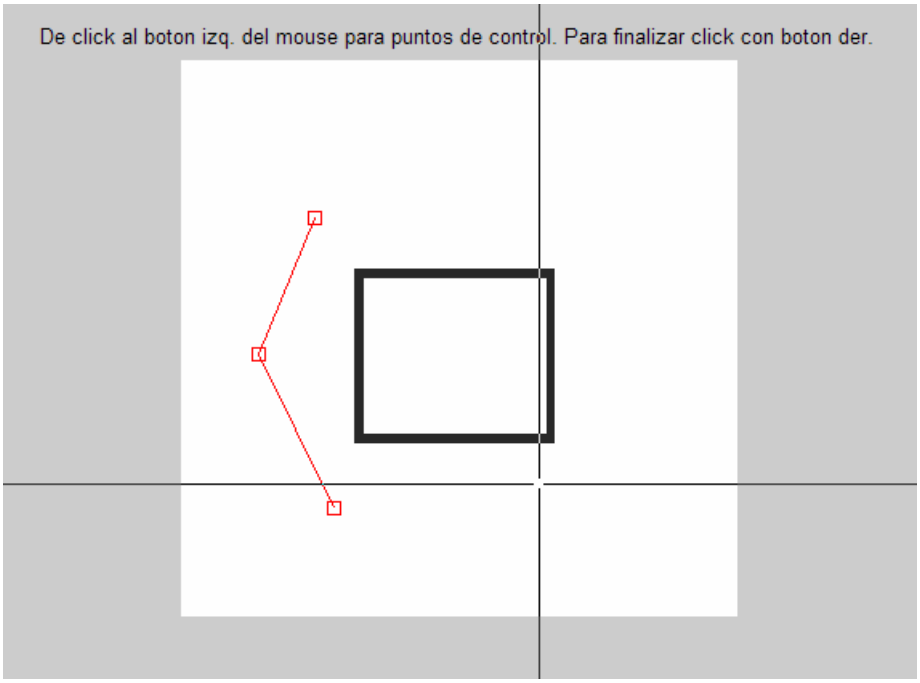

Figura 4.2 *Muestra de la interfase en MATLAB para elegir los puntos de control (cuadros rojos).*

El proceso de deformación del modelo se muestra en la figura 4.3 en donde las variables de entrada son el punto de control anterior $\mathbf{v}_i^{t-1}$ y la fuerza externa representada como $\mathbf{f}_i^{t-1}$ en donde el término $t-1$ no es relevante pues la fuerza externa es considerada constante, como se comentó en el capítulo 2. La matriz pentadiagonal $\mathbf{A}$ debe de tener las características comentadas en el capítulo 2 para que su inversa $\mathbf{A}^{-1}$ sea numéricamente *fácil* de calcular. El último paso es utilizar los datos obtenidos y utilizar la ecuación 2.52 para obtener un nuevo punto de control.

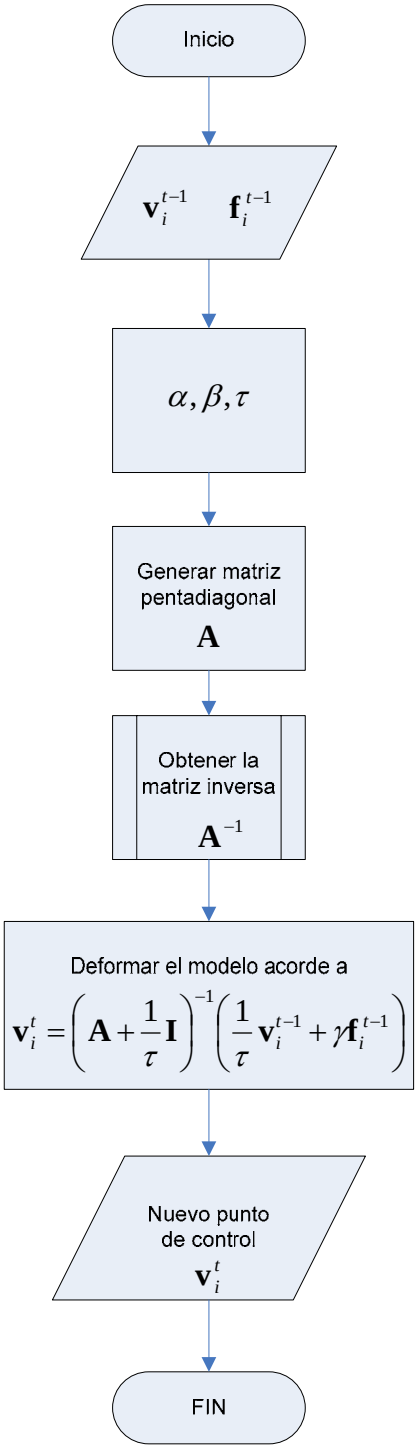

Figura 4.3 *Proceso de deformación para el modelo de Kass*

La figura 4.4 muestra el proceso de ajustar los puntos de control según la distancia entre éstos. El concepto es básicamente calcular la distancia entre los puntos de control y acorde a un umbral definido para la distancia mínima, identificado como **dmin** se observa si la distancia es menor a **dmin**. En caso de ser afirmativo, se quita el punto de control que genere esta distancia mínima. Por otro lado la distancia máxima se fija a un valor llamado **dmax** en donde si la distancia entre los puntos de control sobrepasa este valor de umbral, entonces se agrega un punto intermedio. Con lo anterior se generan o

se quitan puntos de control para que el comportamiento del modelo sea correcto. Cabe resaltar que este proceso es utilizado en todos los modelos.

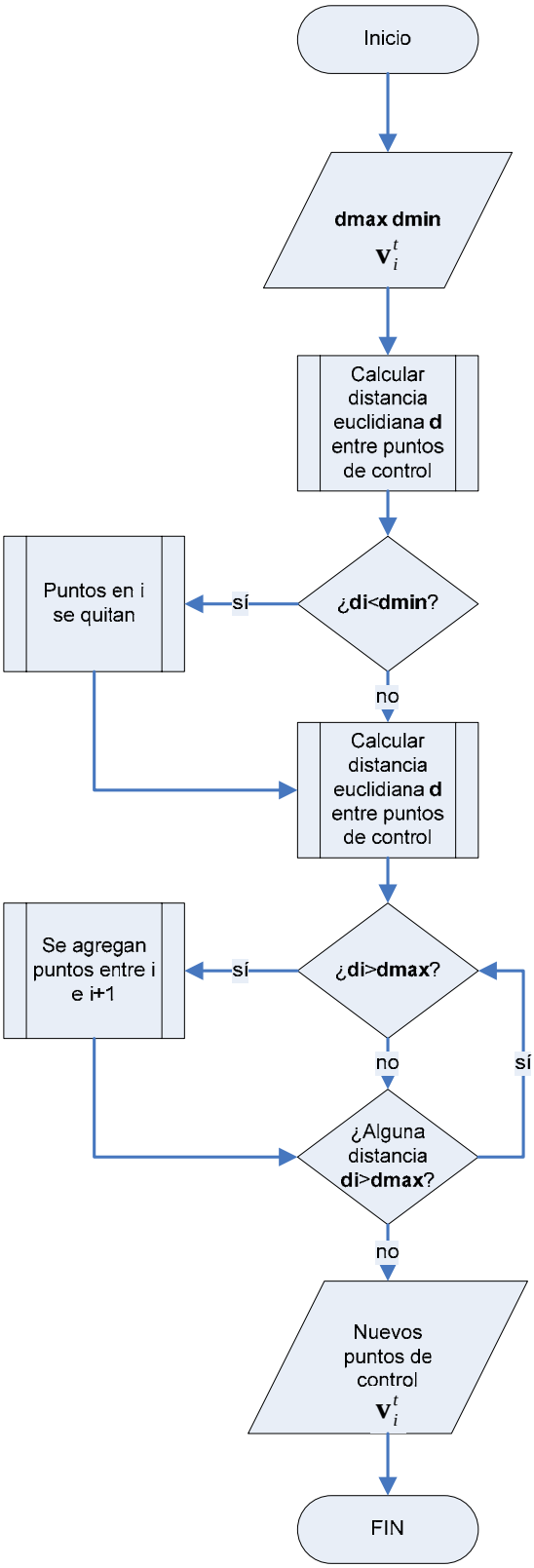

Figura 4.4 *Proceso para ajustar los puntos de control*

4.2 Algoritmo para el modelo de Cohen

Para el modelo de Cohen, la estructura es idéntica a la figura 4.1, sin embargo lo que varía es el proceso en donde se deforma el modelo, ya que se está proponiendo un vector normal para la *fuerza externa*. La figura 4.5 muestra el algoritmo utilizado para deformar el modelo de Cohen. Existen algunos parámetros más que hay que considerar como κ y κ_1, sin embargo la parte relevante en este algoritmo es el cálculo del vector normal $\vec{n}(s)$ que se realiza acorde a la formula del apéndice C. La *fuerza externa* se calcula acorde a la ecuación 3.2 y en base a ella se utiliza la misma ecuación 2.52 variando el término de las fuerzas externas. Al finalizar se tiene el nuevo punto de control.

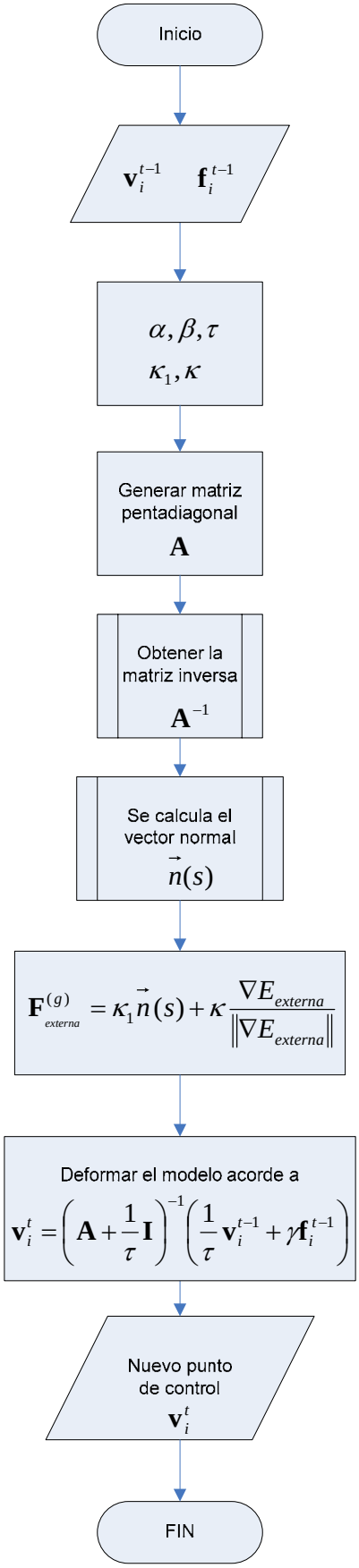

Figura 4.5 *Proceso de deformación para el modelo de Cohen.*

4.3 Algoritmo para el modelo de Xu

Para finalizar, el algoritmo para el modelo de Xu es muy parecido al de Kass (figura 4.1), sin embargo hay que calcular el campo GVF por lo que se propone el algoritmo de la figura 4.6, que presenta los mismos pasos que la figura 4.1 excepto el proceso del cálculo para la obtención del campo GVF. Este proceso se muestra en la figura 4.7 en donde se tienen como entrada; El funcional de energía externa $E_{externa}$, el parámetro μ para el campo GVF y el número de iteraciones N necesarias para generar el campo GVF. Con el funcional de energía externa se calcula el mapa de bordes $f(x,y)$ según la ecuación 3.4 y de éste se obtiene su gradiente para obtener el vector $\mathbf{v}_{GVF}$. A partir de este punto se utilizan las ecuaciones 3.9a y 3.9b para obtener el campo GVF que es la salida final del algoritmo de la figura 4.7.

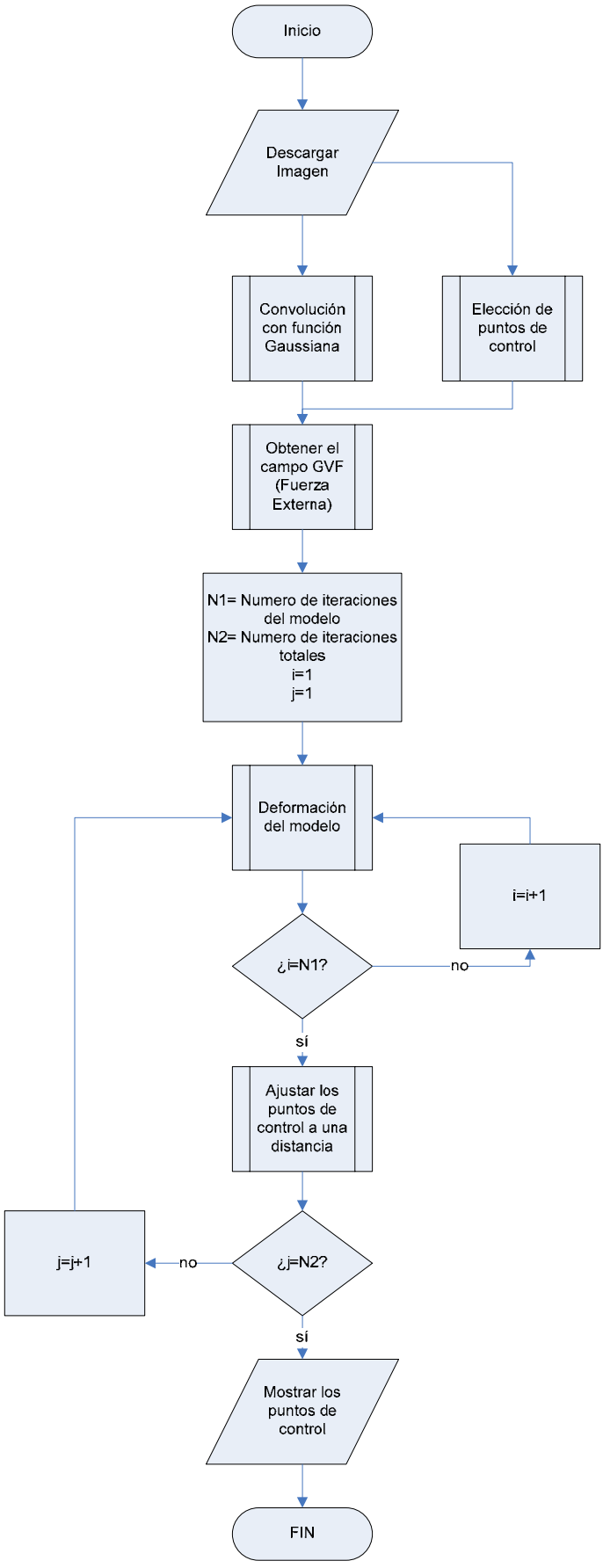

Figura 4.6 *Algoritmo para el modelo de Xu*

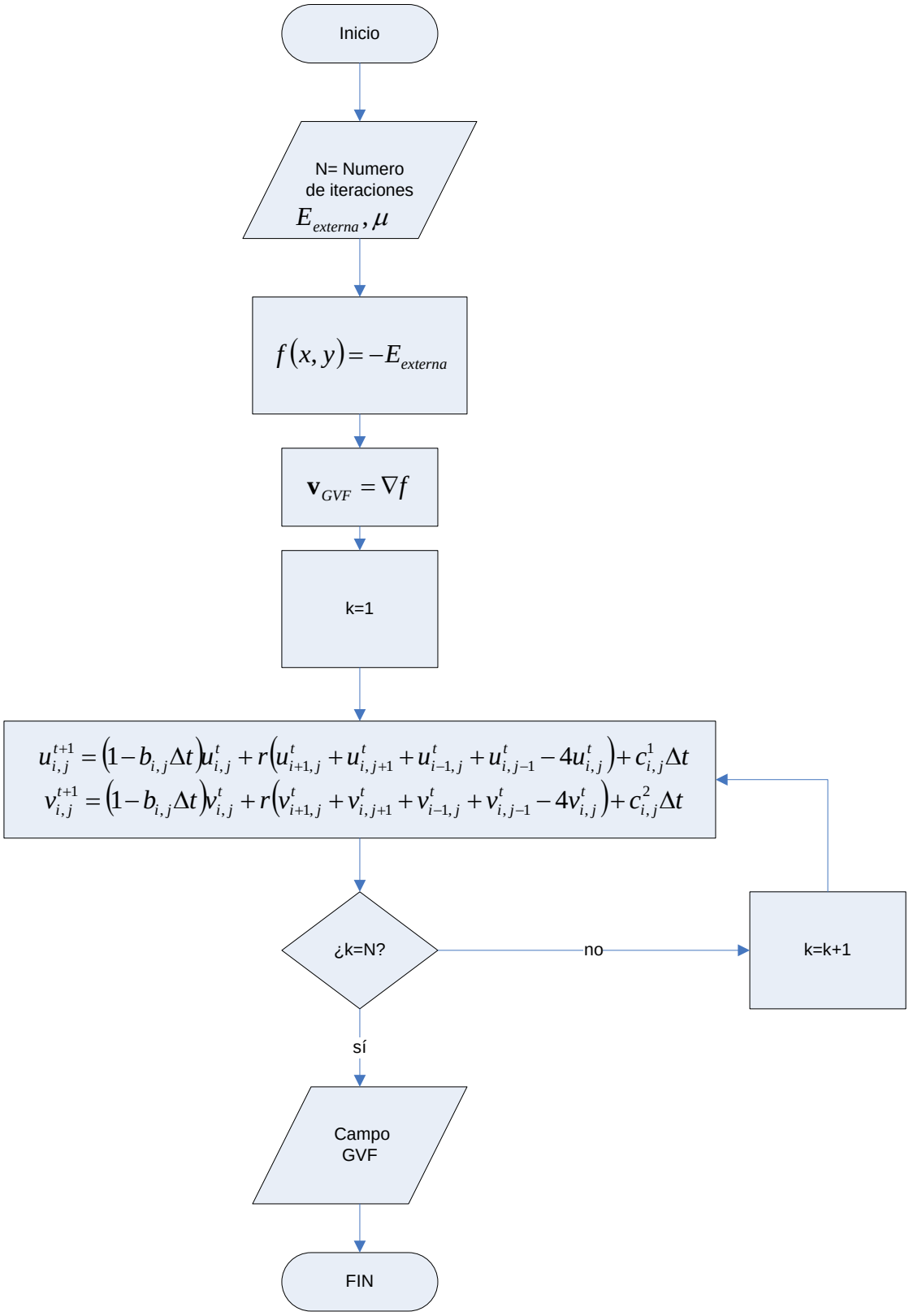

Figura 4.7 *Proceso de deformación para el modelo de Xu.*