UNIVERSIDAD AUTÓNOMA DE YUCATÁN

FACULTAD DE MATEMÁTICAS

TESIS:

Métodos de Conjuntos de Nivel para la Evolución Interactiva de Superficies Implícitas en Entornos Virtuales de Sistemas Háptico

Que para obtener el grado de:

Maestro en Ciencias Matemáticas

Presenta:

Josué Tadeo Moreno Vázquez

Director de tesis:

Dr. Carlos Brito Loeza

Índice general

1. Introducción **5**

1.1. La escultura virtual 5

1.2. El sentido háptico 6

1.3. Antecedentes 8

1.4. Hipótesis 11

1.5. Objetivos 11

2. Representación Implícita de Interfaces **12**

2.1. Introducción 12

2.2. Un poco de topología 13

2.3. Representación Explícita 19

2.4. Representación Implícita 21

2.4.1. Funciones Implícitas 21

2.4.2. Estructuras definidas a partir de la Representación Implícitas 24

2.5. Representación Implícta a través de Funciones Distancia con Signo 26

2.5.1. Funciones distancia 27

2.5.2. Funciones Distancia con Signo 31

2.6. Discretización de Interfaces y estructuras 35

3. Renderización háptica **40**

3.1. Detección de colisiones 41

3.2. Respuesta de colisión 43

3.3. Implementación de la renderización háptica 45

4. Métodos de conjuntos de nivel **48**

4.1. Ecuación de conjunto de nivel 48

4.2. Campos de velocidades 49

4.3. Discretización de la ecuación de conjunto de nivel 50

4.4. Reinicialización 52

5. Modelo de deformación local con volumen constante **54**

6. Resultados experimentales y conclusiones **60**

6.1. Renderización háptica 60

6.2. Deformación local 70

6.3. Compensación de volumen 90

Capítulo 1

Introducción

1.1. La escultura virtual

La tecnología ha avanzado en las últimas décadas a pasos agigantados, permitiendo por un lado utilizarla como una herramienta para realizar más eficientemente las actividades más simples del día a día o hasta las de gran complejidad en cada institución, y por otro lado como una fuente transformadora de nuestro estilo de vida y del funcionamiento de las sociedades; adentrándose en la casa, en el trabajo, en la ciencia, en el arte, en el hombre mismo, ..., y se prevé que en el futuro la realidad virtual sea un mundo paralelo al mundo físico, con el cual se pueda experimentar de igual forma con todos los sentidos, pudiendo almacenar información de cualquier tipo (audiovisual, táctil, etc.), recrear fielmente fenómenos físicos o mejorar procesos de actividades reales.

Un ejemplo motivador de cómo puede la tecnología mejorar actividades realizadas en el mundo real dentro de un mundo virtual, es el proceso de escultura. Esta tarea a grandes rasgos consiste en tener una cantidad suficiente de materia prima para darle la forma que se desea, con una herramienta o con las propias manos. Generalmente empezando con formas básicas simples y terminando con formas más complejas. La variedad de materiales utilizados va desde los muy rígidos, como la madera o la piedra, hasta los más maleables como la cera, la plastilina, el barro, entre otros. Desde hace algún tiempo se han hecho esfuerzos por simular está actividad por computadora, creando con esto los llamados sistemas de escultura virtual.

Se pretende que con esta recreación virtual haya beneficios: el material sería virtualmente infinito, no mancharía, ni sería tóxico, no haría falta espacio para almacenamiento, no habría problemas con las propiedades físicas del material que puedan afectar el trabajo, como que se derrita, se resquebraje, se endurezca; el artista no estaría en contra reloj para terminar su trabajo, podría dejarlo y retomarlo cuando quisiera, la obra se mantendría tal y como se dejó para ser modificada posteriormente. El trabajo no sufriría desgaste ni se destruiría por algún accidente tan fácilmente, sería más fácil deshacer errores, escalar el material para poner detalles, dividir la obra por partes y luego integrarlas en una unidad, entre

muchas otras posibilidades. Esto sería beneficioso para el diseño de autos, la construcción de maquetas, para la industria del cine y la animación, para el arte y muchas otras áreas que la podrían aprovechar.

Estos sistemas consisten de un material que se moldea y de una herramienta con la que se le dará forma, con ambos objetos virtuales. La herramienta se corresponde con un objeto físico externo, que puede ser un ratón, con el que la manipula un usuario. De esto se puede distinguir dos aspectos: el hardware y el software del sistema de escultura virtual.

Se han tenido dificultades en los sistemas de escultura virtual, tanto en los dispositivos físicos como en el software. Por ejemplo, muchos sistemas no son adecuados para la interacción humano-computadora, son poco intuitivos o difíciles de utilizar, como cuando se manipula plastilina, por otro lado la modelación e implementación virtual no son eficientes o precisos. Los dispositivos de entrada más comunes para interacción sólo cuentan con dos grados de libertad, sólo pueden moverse en dos dimensiones sobre una superficie como en el caso del ratón, o simplemente apretando botones como en el teclado, mientras que en el mundo real se cuenta con un espacio en tres dimensiones y con la libertad de movimiento en cualquier dirección; por esto, se han estado desarrollado dispositivos que permitan tanto traslaciones como rotaciones en el espacio, pudiendo hacer combinaciones de movimiento más complejas. Una limitación en la parte de la interacción, es el no poder sentir tactilmente los objetos tridimensionales que están dentro de la pantalla; en el mundo real, sentir los objetos facilita su ontrol y manipulación. Actualmente existen ramas de la computación y la robótica que tratan de integrar el sentido del tacto, además del audiovisual, a algunos sistemas computacionales y robóticos. Para esto se deben tomar en cuenta tanto los componentes físicos del sistema así como los algoritmos que crean el mundo virtual.

1.2. El sentido háptico

Háptica, es una palabra poco conocida hasta el momento, viene del griego *hapesthai*, que significa "*relativo al sentido del tacto*"[1]. En los libros de texto mencionan que existen 5 sentidos: vista, oído, gusto, olfato y tacto; pero investigadores como Klatzy y Lederman [2] han descubierto que en lo que llamamos tacto se incluyen varios sistemas. Estos son el sistema cutáneo, cuyos receptores sensitivos se encuentran en la piel, el sistema kinético, el cual tiene sus receptores en los tendones, articulaciones y músculos, y el sistema háptico, que engloba a los dos sistemas anteriores cuando ocurren al mismo tiempo. Textos como en la referencia [3], definen la palabra háptica como el sentido que abarca a todos los demás que se excluyen en los cuatro sentidos diferentes al tacto, como la percepción del calor, la percepción táctil (como sistema cutáneo), la percepción kinética (percepción de fuerzas o sistema kinético), la nocicepción (proceso neuronal por el cual se procesa las sensaciones nocivas y el dolor) y parte de la percepción del equilibrio. También se le llama háptica a la subrama de la sicofísica que estudia el sentido háptico, fundada a principios del siglo XX [4]. De esta forma el concepto de háptica ha adquirido un significado más amplio que el de la palabra tacto.

El sentido háptico nos ayuda a detectar y reconocer objetos y superficies. También ayuda en la realización de tareas que suponen resistencia de fuerza como al utilizar cualquier tipo de de herramienta, al sostener una taza de café, al cortar carne, al escribir, al escoger la fruta en el supermercado, al pintar o al esculpir una obra de arte. Esto es debido a que las personas son capaces de desplegar y percibir pequeñas variaciones de resistencia de fuerza, lo que hace que ellas tengan control sobre sus movimientos y sobre los objetos que manipulan; como dicen Maclean y Hayward [5], "*Para los humanos, precisión requiere resistencia*". Además, la háptica complementa a los otros sentidos, con ella podemos percibir/reconocer ciertas propiedades de los objetos y superficies que con los demás serían imprecisas de determinar o difíciles de reconocer, tales como la textura, la rigidez o el peso [2], así como la temperatura y la presión cuando se tiene contacto con un objeto.

Por otra parte, la tecnología actual está llena de objetos que estimulan nuestra vista y nuestros oídos, pero casi no se le ha dado importancia al sentido del "tacto "a pesar de su importancia en muchas actividades que realizamos. Sin embargo, hace algún tiempo, entre las décadas de 1970 y 1980, la investigación háptica fue centro de atención de otros campos de investigación, como la robótica, cuando empezaron a interesarse en la manipulación y la percepción [4]. En consecuencia han surgido términos ampliamente utilizados dentro de la computación y la robótica como los siguientes [3]:

- **Interacción háptica**: describe la transmisión de información háptica uni- o bidireccionalmente.
- **Dispositivo háptico**: es un sistema de entrada-salida que genera "señales"percibidas por el sentido háptico.
- **Interfaz háptica**: es un dispositivo háptico que permite interacción háptica, mediante el cual la información transmitida es sujeta a cambios y es medida de la interacción. Se refiere a los datos y al dispositivo háptico.
- **Renderización háptica**: es el proceso de representación (o simulación) de la información háptica hecho por una computadora.
- **Retroalimentación háptica**: se refiere a la información transmitida por interacción kinética, en este caso conocido también como retroalimentación de fuerza, o también táctil, la que se produce con la interacción de la piel.

Aunque el concepto de háptica es amplio, en el argot computacional, el término de renderización háptica, generalmente, como lo expresa el último concepto de la lista anterior, se entiende como la simulación de fuerzas producidas por la presión o de la fricción cuando se tiene contacto con un objeto; lo que recrea la sensación de rigidez y de textura, lo que a su vez crea la ilusión de sentir.
Uno de los aspectos interesantes del sentido háptico, es el hecho de que no es necesaria la experiencia por contacto directo de las partes de nuestro cuerpo para sentir el objeto que se manipula. Ya que a través de una herramienta se pueden percibir sus características hápticas [6], al igual que se hace con las

manos. Este fenómeno, uno lo puede detectar, por ejemplo, cuando se está en un coche. Éste se convierte en una extensión del cuerpo y se puede sentir casi cualquier golpe o contacto a su alrededor, como las vibraciones debidas a la regularidad/irregularidad de la superficie sobre la cual se transita. En el caso de objetos virtuales el cerebro y el cuerpo los detecta como reales, según la calidad de la simulación y la forma de interacción usuario-máquina.

El sentido haptico es especialmente sensible, por lo que al recrear la sensacion de tacto para los objetos de un ambiente virtual es necesario que el cálculo de las fuerzas sea rápido y congruente con la visualizacion del ambiente virtual, ya que un desface entre los dos sistemas es percibido con facilidad. Para esto se requiere que la velocidad de la renderizacion háptica sea incluso más rápido que la renderizacion visual, al menos de 1000 *hertz* ($1kHz$) [42]. Esto resulta desafiante, cualquier cálculo complejo debe ser simplificado lo más posible sin perder estabilidad.

1.3. Antecedentes

La parte virtual de un sistema de escultura requiere de varios algoritmos y conceptos para poder funcionar. Entre los aspectos más importantes, uno es la construcción del objeto al que se le dará forma (su geometría), y otro es el enfoque que se le da a la actividad de moldeamiento. Primero, el tipo de ambiente virtual en los sistemas de escultura es tridimensional, y el material a deformar puede ser una superficie bidimensional [14], por ejemplo una hoja de papel, o un objeto tridimensional. El más común es el del último estilo, para el cual los programas tienen un algoritmo para su visualización, que construyen visualmente el objeto completo a partir de formas simples denominadas *primitivas* como puntos, rectas, polígonos e incluso poliedros, o combinaciones de éstos, incluyendo aquellas cuyas superficies están particionadas por una malla de figuras poligonales (comúnmente regulares, en específico triángulos).

Para que se implemente la visualización, se necesita definir los puntos y la forma en que ellos están conectados, esta puede ser hecha a mano, conectando uno a uno todos los puntos relacionados o contiguos, y formar primitivas de un orden superior (como líneas o triángulos), que sería impráctico desde el punto de vista de escultura, aunque de esta forma los objetos se crearían como un dibujo, pero éste no es el objetivo. En computación se utilizan diferentes tipos de estrategias o representaciones, para definir el objeto tridimensional (o bidimensional) de una manera más eficiente, y en una aplicación puede emplearse un tipo o una combinación de ellos; algunas incluyen estructuras matemáticas como funciones y ecuaciones. Existen dos categorías principales de representación: las *representaciones de fronteras* (B-*rep*, por *boundary representation* en inglés) y las *volumétricas* (V-*rep*, por *volume representation*). Entre las primeras se encuentran *superficies de subdivisión* [45] y *mallas poligonales* [29], que solamente definen la frontera del objeto, y entre las segundas están las *superficies implícitas*[43], los *sólidos de subdivisión* [30] y modelos basados en *descomposición celular* [18] (donde *celular* hace referencia a los *vóxeles* unidades de volumen con los que son formados los objetos 3D, o incluso *píxeles* para objetos 2D), que son representa-

ciones que incluyen el interior. Algunas de estas representaciones tiene sus propias variantes, según cómo son construidas dichas funciones, unas emplean funciones analíticas como B-*splines* para representación implícita [14] y para representación explícita [23], en este último caso también existen los B-*Splines no uniformes* [24] (NURBS, por sus siglas en inglés) y puede utilizarse un método denominado B-*Spline jerárquico* [25]. De todas ellas se pueden hacer híbridos o crear otras categorías de representación.

En casos volumétricos a veces se emplean estructuras de datos en la que son organizados las discretizaciones de los objetos. Aunque se puede crear toda una malla uniforme cúbica para definir el modelo como en [26], también existen otras estructuras útiles: en [20, 28] es utilizado un *árbol binario* para ahorrar memoria computacional al guardar los datos de la representación, otros manejan una estructura de multirresolución *octree* como [18, 19] que pueden dar gran nivel de detalle.

Dentro del ambiente virtual también varía mucho el enfoque dado a la deformación del objeto. El objetivo principal, es que la actividad sea lo más natural e intuitiva posible, concentrando al usuario más en la deformación del objeto que a la estructura matemática subyacente al hacerlo. Se podría decir que unas herramientas virtuales primitivas son los cursores o las herramientas puntuales sin tamaño ni forma [15] (interacción punto-objeto). Algunos utilizan herramientas tridimensionales (interacción superficie-objeto), típicamente esferas y elipsoides [20, 28] aunque pueden tener formas arbitrarias [18, 20, 28], las cuales también cuentan con su propia representación, que puede ser implícita [14, 20, 41]; tienen su forma particular de interactuar: algunos cambian los valores de la función implícita que representa al material deformable con la intersección de la herramienta [20, 26, 41], otros aplican un campo vectorial, hay casos en que se utilizan ecuaciones diferenciales para el movimiento [27], geometría sólida constructiva (CSG), representaciones físicas dinámicas [14, 23, 24], o representaciones de material [18] como dureza, densidad, entre otras, que se van modificando según la intervención de la herramienta; y hay diferentes operaciones a realizar, como operaciones booleanas, doblar partes, tallar, indentar, imprimir formas, separar, fusionar o extraer material, abultar, . . . , e incluso pintar [18]; también se puede tener cierto grado de movimientos rígidos: torsiones, rotaciones y traslaciones. Existen sistemas de escultura que cuentan con una diversidad de herramientas, las cuales muchas no tienen equivalente en el mundo real como en [27]. Entre otros aspectos, algunos sistemas como [14], le dan mucha importancia a la forma y tamaño de la herramienta; en este caso particular se utiliza una estructura adicional de representación para la deformación dinámica de su superficie, representada geométricamente con B-splines, cuyos elementos se denominan como *puntos de masa*, esta técnica es conocida como *Sistema de Masa-Resorte* (MSS, por susu siglas en inglés), cuya integración con puntos de la representación geométrica fue hecha por primera vez por Dachille et al [15].

Hay sistemas de escultura que integran dispositivos físicos no convencionales, como los dispositivo que amplían la libertad de movimiento y que incluso pueden dar retroalimentación háptica [14, 18, 22, 25, 27, 37, 41]. Ahora bien, existen sistemas de escultura háptica que pueden generar diferentes tipos de fuerzas, según sus al goritmos y la configuración de su dispositivo físico, como las táctiles, inclusive fuerzas de fricción [41], o también fuerzas de torque [16], para figuras con tres dimensiones. Kim [17] introdujo un

algoritmo háptico basado en una representación de superficie implícita, tal que los valores de las funciones que las representan fueron discretizados en una red regular tridimensional, que permite sentir al usuario una superficie suave sin discontinuidad de fuerza.

Los sistemas de escultura virtual, desde el punto de vista de usabilidad deberían ser estables, poder obtener una forma deseada, que sean fáciles de usar, con una interacción natural, simple e interactiva. Deberían tener el mismo nivel de libertad de movimiento que en el mundo real, deberían tener capacidad de incluir gran nivel de detalle, las modificaciones deben ser hechas en tiempo real, poder crear puntas y detalles, así como que las herramientas no traspasen visualmente el material. Desde el punto de vista computacional deberían ser eficientes en cuanto a tiempo y memoria. Cuando se integra renderización háptica, además de la rapidez con lo que deben ser efectuadas, debe haber una correspondencia entre la geometría y las fuerzas calculadas, estos campos de fuerzas deben ser continuos, por ejemplo, al haber material visual debe sentirse, y viceversa, tampoco debe haber cambios bruscos en dirección, pero sí representar las puntas, o bordes, agudos o afilados, así como la posibilidad de detectar con la herramienta virtual objetos delgados o pequeños. Algunos sistemas tratan de emular diferentes tipos de materiales, ya sea un objeto rígido como la madera virtual de [21], o un objeto maleable [18, 20, 28], que son la mayoría. Una metáfora de escultura especial es la denominada arcilla virtual (*virtual clay*) [30, 31, 35]. El objetivo de éstos, es que el material utilizado se parezca precisamente a la verdadera arcilla, que exhibe un estado material entre un fluido viscoso y un sólido plástico [31]. Algunas características básicas que se requieren para esto son: posibilidad de cambios topológicos, como crear orificios (como el de una dona), tapar orificios, poder partir y pegar, pedazos o partes del mismo material; posibilidad realizar deformaciones locales, hacer deformaciones globales y evitar auto-intersecciones de partes del material (como si una parte estuviera dentro de la otra). La representación volumétrica parece ser una buena elección para hacer cambios de topología fácilmente. Sin embargo el material puede reducirse, es decir, la cantidad de material (o volumen del objeto tridimensional) no se mantiene constante, teniendo en cuenta que no se agrega ni se quita material al trabajar. Una solución propuesta ha sido utilizar *autómatas celulares* [32, 33] para realizar deformaciones locales en arcilla virtual. Como es lógico y natural pensar, se pueden utilizar modelos físicos para una interacción más real. Hay basados en modelos de sólidos plásticos como [25]. Sin embargo, debido a que el costo computacional en tiempo de la implementación física, es alto, sobre todo para objetos geométricos con formas complejas y aún más con renderización háptica, la actividad de escutura se vuelve poco o nada interactiva; son necesarias simplificaciones, herramientas o trucos computacionales, como precalcular jerarquías [24], para poder alcanzar manipulación en tiempo real. En un futuro esto podría ser solventado con el incremento de la velocidad de procesamiento de las computadoras o por la programación en paralelo.

Ahora bien, existen ciertas técnicas conocidas como *métodos de conjuntos de nivel* que han sido utilizados para modelar comportamientos tales como la de una flama en una combustión, de sólidos que parecen fluidos o la interacción de fluidos con diferentes densidades [36]. Estos métodos hacen uso de

técnicas numéricas precisas y robustas para modelar movimientos complejos de superficies y curvas [38]. Desde 1988, introducidos por Sethian y Osher [39], han sido aplicados en muchas ramas científicas como en mecánica de fluidos, en procesamiento de imágenes, en combustión, visión computacional, entre otros. [40]. De hecho también ha sido aplicado a escultura virtual. En [43] se utiliza una función distancia con signo y también diferentes campos de velocidad para modificar los valores de la función distancia con signo, y definen la forma de interactuar de las herramientas. Otro sería [44], el cual las herramientas son curvas o trazos sobre la cual se implementa un campo velocidad para modificar un objeto representado implícitamente. Sin embargo ninguno de los dos trabajos se conserva la propiedad de mantener el volumen del objeto, por lo cual sería un factor interesante para investigar.

1.4. Hipótesis

Los métodos de conjuntos de nivel proporcionan un mecanismo adecuado para la evolución de una superficie implícita, ya que han demostrado facilitar el manejo de cambio de forma de figuras, pudiendo incluso cambiar la topología. Entonces, esto permitirá simular la deformación de un sólido para adquirir una estructura deseada de manera interactiva, cuya actualización de manipulación por parte de un usuario sea cercano al tiempo real, sin variación de cantidad de material (sin cambio de volumen), tal que pueda ser incorporada renderización háptica de una manera eficaz y eficiente, sin problemas de discontinuidades o inestabilidad, acoplado con la renderización virtual tanto en espacio como tiempo.

1.5. Objetivos

- Implementar una forma de representación y deformación local de una superficie implícita simulando un objeto deformable, para llevar a cabo la evolución de la misma sin que cambie el volumen total de la interface, con el uso de los métodos de conjunto de nivel.
- Integrar una forma de renderización háptica a un objeto representado implícitamente, sin aplicar deformación, haciendo uso de sus conjuntos de nivel para llevar a cabo tanto el módulo de detección de colisión como el de respuesta de colisión.

Capítulo 2

Representación Implícita de Interfaces

2.1. Introducción

En la inmensa mayoría de los objetos que existen en el mundo tridimensional se puede observar la determinación de dos espacios, uno, el ocupado por el objeto y, el otro, el ambiente externo. También está la parte del objeto que se puede identificar desde su exterior, la más superficial, y el interior de éste consiste del volumen que ocupa su masa. No importa que el objeto tridimensional cuente con agujeros o concavidades, que esté hueco e incluso si el objeto es tan plano como una hoja de papel que su interior no parezca tener volumen, siempre es posible verlo de esta forma. Así también, objetos tales como la arena o la azúcar, pueden considerarse desde la perspectiva de un solo objeto o como la reunión de una multitud de partículas, que al considerarlas individualmente determinan regiones como las mencionadas anteriormente. Si uno se abstrae un poco y se mueve a un mundo bidimensional como el plano, algo similar ocurre con algunos objetos que viven ahí, sólo que la parte que los limita es un borde y su interior no tiene volumen sino área. Por tanto, para modelar objetos bidimensionales y tridimensionales computacionalmente, al menos para su visualización, se hace uso de la frontera que limita a estos objetos de su exterior, esto es, se utilizan objetos abstractos como curvas y superficies para definirlos.

Aunque a veces sólo se utilicen medios puramente alámbricos o de volumen para manualmente construir el modelo gráfico de un objeto, dando puntos y conexiones entre ellos por medio de líneas curvas o rectas, además que pueden haber ambigüedades, es importante en muchas ocasiones contar con información local o global del modelo, como la suavidad entre las uniones de los puntos, o ser muy precisos en la expresión gráfica de estos objetos. Por tanto, se necesita utilizar expresiones matemáticas que describan o se aproximen a las curvas y superficies que forman a los objetos de interés. Para esto, es importante primero que nada, la forma en que se define a una curva o superficie particular, es decir, su representa-

ción matemática, ya sea explícita o implícita. Según la representación, se podrán realizar determinados cálculos, incorporar propiedades físicas a los modelos, llevar a cabo el despliegue visual, entre otras cosas.

En este capítulo se hablará de forma general sobre los tipos de representación que existen, concentrando interés especial en la representación implícita. Al final se introduce el concepto de *función distancia con signo*, que es una herramienta importante para modelar la evolución de una curva o superficie por medio de los métodos de conjuntos de nivel, debido a sus propiedades particulares. Pero antes, se trata de definir los tipos de curvas y superficies que modelarán a los objetos, prescindiendo del tipo de representación, haciendo mención de sus propiedades topológicas mas importantes.

2.2. Un poco de topología

El objetivo de esta sección, en principio, es definir de manera topológica a las curvas en $\mathbb{R}^2$ y a las superficies en $\mathbb{R}^3$ que servirán para modelar a un único objeto, según sea el caso, bajo la perspectiva de que cada objeto divide al espacio ambiente en tres partes: el interior, que es el espacio que ocupa, el exterior, su espacio circundante, y su superficie, la frontera que delimita a dicho objeto de su exterior. También se pretende dar a conocer cuántos y cuáles son los tipos existentes de estas curvas y superficies así como enunciar sus propiedades más importantes. Entonces, en esencia se tratará de contestar las siguientes preguntas: ¿Qué es una curva?¿Qué es una superficie? ¿Qué clases de curvas y superficies son utilizadas para modelar los objetos?¿Cuántos tipos de estas curvas y superficies hay?¿Qué propiedades cumplen? En el apartado anterior se mencionan objetos huecos, sin embargo, si no es importante expresar esta característica, al modelarse computacionalmente puede ser representado por la superficie "más exterior", en caso contrario entonces consistiría de dos superficies (disconexas) y parte del espacio que rodea el cuerpo es en realidad exterior a él; en primera instancia este caso no se considerará, pero más adelante se tomará en cuenta. Antes de empezar cabe aclarar que esta parte sólo pretende ilustrar el modelado de cuerpos (tanto de dos como de tres dimensiones) desde un enfoque abstracto como es la topología, sólo para dar una idea hasta que nivel puede llegar este problema.

Formalmente, la topología es la parte de la matemática que estudia las propiedades de los espacios topológicos[1] que se mantienen bajo funciones continuas[2]. Desde una persepectiva más informal, trata de estudiar las propiedades que conservan los conjuntos de puntos cuando son deformados sin pegarse ni romperse; la idea es más intuitiva cuando el conjunto considerado es una curva o una superficie. Esta

[1] Un espacio topológico es un par ordenado (X, τ), donde X es un conjunto y τ es un conjunto de subconjuntos de X que cumple: **1)** $\varnothing, X \in \tau$, **2)** si $A_1, A_2 \in \tau$, entonces $A_1 \cap A_2 \in \tau$, **3)** si $\mathcal{A} \subset \tau$, entonces $\bigcup_{A \in \mathcal{A}} A \in \tau$. Al conjunto τ se le conoce como topología, a sus elementos como *conjuntos abiertos*, y a los complementos de los abiertos se les llama *conjuntos cerrados*. Se puede decir simplemente que X es un espacio topológico si no hay riesgo de confusión sobre la topología τ con la que es dotado al conjunto X.

[2] Una función continua $f : (X, \tau_1) \longrightarrow (Y, \tau_2)$, entre espacios topológicos (X, τ_1) y (Y, τ_2), cumple que para cualquier abierto $A \in \tau_2$, entonces la imagen inversa $f^{-1}(A) = \{\ x \in X : f(x) \in Y\} \in \tau_1$. Esta definición es equivalente a la dada con límites en cálculo de varias variables.

rama matemática permite una definición intrínseca general de curvas y superficies sin la necesidad de encajarlas en algún espacio ambiente y sin hacer alusión a ningún tipo de representación; si bien lo primero es cierto, en este caso sólo se considerarán curvas y superficies inmersas en algún espacio $\mathbb{R}^m$. Para poder entender lo siguiente, es necesario conocer los conceptos más básicos, como el de topología, espacio topológico, función continua, homemorfismo, espacios homemorfos, entre otros [46, 47]

En este punto se empezará por introducir una lista de términos y notaciones sobre algunos conjuntos especiales relacionados a cualquier conjunto de un espacio topológico.

Definición 2.2.1 *Sea (X, τ) un espacio topológico y $A \subset X$. Entonces:*

1. *El interior de A es la unión de todos los abiertos que están contenidos en A, esto es,*

$$\text{int}A = \bigcup_{\substack{U \in \tau \\ U \subset A}} U. \tag{2.1}$$

2. *La frontera de A está definida como los puntos de X cuyos abiertos siempre tienen puntos de A y puntos de su complemento,*

$$\partial A = \{\ x \in X :\ \textit{cualquier } U \in \tau \textit{ que contenga a } x \textit{ debe cumplir } U \cap A \neq \varnothing \neq U \cap A^c\ \}. \tag{2.2}$$

3. *La clausura o cerradura de A es la unión del interior con la frontera,*

$$\overline{A} = \text{int}A \cup \partial A. \tag{2.3}$$

4. *El exterior de A es el complemento de la clausura,*

$$\text{ext}A = \overline{A}^c. \tag{2.4}$$

Estos conceptos coinciden con la noción que se tiene de interior, exterior y frontera de un conjunto que forma un cuerpo sólido en el espacio o de un objeto bidimensional en el plano. Algunas propiedades a destacar sobre estos conjuntos son, que el interior y el exterior de cualquier conjunto son conjuntos abiertos, mientras que la frontera y la clausura son conjuntos cerrados. Hay otras relaciones que se destacan en la siguiente proposición.

Proposición 2.2.2 *Sea (X, τ) un espacio topológico y $A \subset B \subset X$, entonces:*

1. $\text{int}A \subset A \subset \overline{A}$, $\overline{A} \subset \overline{B}$, $\text{int}A \subset \text{int}B$, $\partial A = \partial A^c$ *e* $\text{int}A^c = \overline{A}^c$.

2. *Si A es abierto,* $\text{int}A = A$.

3. *Si A es cerrado,* $\overline{A} = A$.

Los términos antes descritos serán utilizados más adelante, pero por ahora se introducirá uno de los conceptos más importantes de la topología, el de *homemorfismo*. Éste determina qué espacios son idénticos en el sentido de tener las mismas propiedades topológicas.

Definición 2.2.3 *Dados dos espacios topológicos (X, τ_1) y (Y, τ_2), un homemorfismo entre ellos es una función continua e invertible $f : X \longrightarrow Y$, con inversa continua. Si existe tal función f se dice que X y Y son homeomorfos, denotando a esta relación por $\cong$, es decir, $X \cong Y$. También se dice que $M \subset X$ es homeomorfo a $N \subset Y$ cuando son homeomorfos como espacios topológicos con las topologías subespacio*[3] *correspondientes.*

Ahora, se definen ciertos espacios topológicos denominados como *variedades*, que localmente son muy parecidos a $\mathbb{R}^n$, y en base a ellos se definen las curvas y superficies. Aunque el término de variedad es más general, aquí se considerarán solamente subespacios de espacios topológicos $\mathbb{R}^n$ dotados con la topología usual[4].

Definición 2.2.4 *Sea M subespacio de $\mathbb{R}^n$ con la topología usual. Se dice que M es una variedad m-dimensional o m-variedad (con $m < n$), si para cualquier elemento $\mathbf{x} \in M$, existe un conjunto abierto $\mathcal{U} \subset \mathbb{R}^n$ que contiene a $\mathbf{x}$ y que es homeomorfo a un abierto $\mathcal{V}$ de la topología usual de $\mathbb{R}^m$.*

Esta definición es la usual para variedades, que también suelen llamarse *variedades sin frontera* o *sin borde*, para distinguirlas de aquellos espacios que cuentan con puntos que tienen *vecindades*[5] muy parecidas a conjuntos abiertos del *semiplano superior*[6] del correspondiente espacio $\mathbb{R}^n$, como se ilustra en la Figura 2.1. A estos últimos se les denomina por contraposición como *variedades con frontera*. Hay que aclarar que la frontera de una variedad no es la misma que la frontera topológica de conjunto dada anteriormente, pero podrían coincidir.

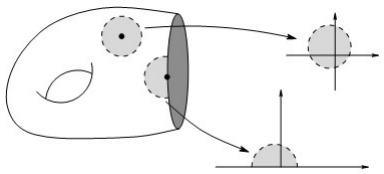


Figura 2.1: Variedad con frontera.

Ejemplo 2.2.5

1. *Un conjunto finito forma una variedad $0-$dimensional. Cada punto de un conjunto finito siempre tiene una vecindad que consta de un solo elemento, que evidentemente es homeomorfo al único abierto no vacío de $\mathbf{R}^0 = \{\ \mathbf{0}\ \}$, él mismo.*

2. *Cualquier conjunto abierto $\mathcal{U}$ como subespacio de $\mathbb{R}^n$ con la topología canónica es una variedad de dimensión n. Ya que para cada punto $\mathbf{x} \in \mathcal{U}$ siempre existe una bola abierta $\mathbb{B}_\epsilon(\mathbf{x})$ de $\mathbb{R}^n$ contenida*

[3] Si (X, τ) es un espacio topológico y $Y \subset X$. La *topología subespacio* inducida por X sobre Y se define como $\widetilde{\tau} = \{\ \mathcal{U} \cap Y : \mathcal{U} \in \tau\ \}$. Al espacio topológico $(Y, \widetilde{\tau})$ se le conoce como subespacio de (X, τ); en este caso se dice que Y es subespacio de X.

[4] La topología usual o canónica de $\mathbb{R}^n$ es la formada por las uniones arbitrarias e intersecciones finitas de bolas abiertas $\mathbb{B}_\epsilon(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^n : ||\mathbf{x} - \mathbf{y}|| < \epsilon\}$ con $\epsilon > 0$ y $\mathbf{x} \in \mathbb{R}^n$.

[5] Una *vecindad* de un punto perteneciente a un espacio topológico es un conjunto abierto que contiene a ese punto.

[6] El conjunto $\mathbb{H}^n = \{\ (x_1, \ldots, x_n) \in \mathbb{R}^n :\ x_n \geqslant 0\ \}$ es conocido como *semiplano superior* de $\mathbb{R}^n$

en $\mathcal{U}$*. Esta bola es un abierto del subespacio formado por* $\mathcal{U}$*, también es un abierto de la topología canónica de* $\mathbb{R}^n$*.*

Antes de seguir, no está demás aclarar que una variedad no puede tener vecindades homeomorfas a abiertos de dos espacios $\mathbb{R}^n$ diferentes, por lo que no se puede ser n-variedad y m-variedad al mismo tiempo si $m \neq n$. Casos como la unión disjunta de un plano y un círculo en $\mathbb{R}^3$ no son considerados variedades, ya que hay puntos cuyas vecindades son homeomorfas a abiertos de $\mathbb{R}^2$ y otros puntos tienen vecindades homeomorfas a abiertos de $\mathbb{R}$. Dada esta aclaración ya se está preparado para la de definición de curva y superficie.

Definición 2.2.6 *Sea* M *subespacio de* $\mathbb{R}^n$ *con la topología usual. Se dice que* M *es una curva o superficie si es una* 1*-variedad o* 2*-variedad, respectivamente.*

A pesar de haberse definido como espacios topológicos, para efectos prácticos las curvas y superficies siguen siendo conjuntos. Ahora bien, se sabe que no cualquier tipo de curva o superficie se utilizará para llevar a cabo la modelación de los objetos. Como se mencionó al principio, se espera que tanto las curvas como las superficies sean el límite de dos regiones, una que represente el interior y otra el exterior de los objetos modelados. Esta propiedad de separación sólo la cumplen un subconjunto de un tipo especial de variedades que se les conoce como *variedades cerradas.*

Definición 2.2.7 *Una* m*-variedad* $M \subset \mathbb{R}^n$ *es cerrada si es un espacio topológico compacto. Si* $m = 1$*, a* M *se le llama curva cerrada; si* $m = 2$*, se le llama superficie cerrada.*

Al ser M un subespacio compacto de $\mathbb{R}^n$, según el *Teorema de Heine-Borel*, es equivalente a decir que el conjunto subyacente M es un conjunto *acotado*[7] y cerrado en $\mathbb{R}^n$. No hay que confundir la definición de variedad (curva, superficie) cerrada con la de conjunto cerrado, ya que son conceptos diferentes.

Una de las características topológicas que tienen las variedades, en particular las cerradas, es que si son de una sola pieza, siempre contienen una curva que une a dos puntos arbitrarios de ellas. A estas propiedades se les conoce, respectivamente, con los nombres de *conexidad* y *conexidad* por caminos, que en general no son conceptos equivalentes, pero la segunda siempre implica a la primera.

Otra propiedad importante que cumplen las variedades topológicas cerradas de dimensión n es que pueden dividir al espacio $\mathbb{R}^{n+1}$ en dos partes, siempre y cuando puedan ser *encajadas*[8] en él [13].

Teorema 2.2.8 (*Teorema de separación de Jordan-Brouwer-Wilder*) *Cualquier* n*-variedad* M *cerrada y conexa encajada en* $\mathbb{R}^{n+1}$*, separa al espacio en dos conjuntos conexos con frontera común al conjunto subyacente* M*.*

[7] Un conjunto es *acotado* en $\mathbb{R}^n$ si está contenido en una bola abierta $\mathbb{B}_\epsilon(\mathbf{x})$.

[8] Se dice que un espacio topológico (Y, τ_1) está *encajado* en (X, τ_2) si existe una función continua inyectiva $f : Y \longrightarrow X$, tal que para el subespacio $f(Y)$ de X, la función $g : f(Y) \longrightarrow Y$ con $g(f(y)) = y$ es una función continua. También se dice que Y está embebido o inmerso en X.

El teorema anterior indica que los conjuntos conexos resultantes son disjuntos y abiertos, por tanto son conexos por caminos. En el caso de $n > 1$, que incluye a curvas y superficies cerradas, uno de los conjuntos conexos es acotado mientras que el otro no, tal como se requiere para modelar objetos.

La pregunta a contestar ahora es: ¿Cuántos tipos de curvas y superficies cerradas existen? Para curvas cerradas conexas, sólo existe una salvo homeomorfismos, el círculo unitario $\mathbb{S}^1$ en $\mathbb{R}^2$, que puede ser encajado en cualquier $\mathbb{R}^m$ con $m \geqslant 2$. Algo similar ocurre con las superficies cerradas.

Primero se ilustrará lo que se entiende por *suma conexa* de dos superficies cerradas conexas. Como se muestra en la Figura 2.2, dadas dos superficies A y B, se les hace un hoyo a cada una de ellas generando como resultado dos bordes, uno en cada una. La *suma conexa* de estas dos superficies, denotada como $A\#B$, es la superficie obtenida al pegarlas por los bordes. Para una definición formal y algunas propiedades básicas de la suma conexa puede verse en la referencia [12].

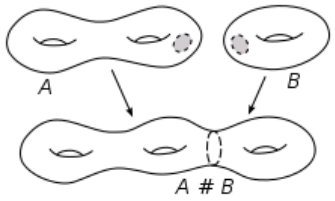


Figura 2.2: Suma conexa de A con B.

Teorema 2.2.9 *Teorema de Clasificación de Superficies. Sólo existen 3 familias de superficies cerradas conexas:*

1. *Superficies homeomorfas a la esfera unitaria* $\mathbb{S}^2$.
2. *Superficies homeomorfas a una suma conexa de un número finito (no nulo) de toros*[9] $\underbrace{\mathbb{T}\#\ldots\#\mathbb{T}}_{n \text{ veces}}$.
3. *Superficies homeomorfas a una suma conexa de de un número finito (no nulo) de planos proyectivos*[10] $\underbrace{\mathbb{P}^2(\mathbb{R})\#\ldots\#\mathbb{P}^2(\mathbb{R})}_{n \text{ veces}}$.

Todas las superficies pertenecientes a la tercer familia no pueden ser encajadas en $\mathbb{R}^3$. Estos viven en espacios $\mathbb{R}^n$ con al menos $n = 4$, por tanto no separan al espacio en dos partes como lo dice el **Teorema 2.2.8**, por lo que no son de interés. En cambio, las superficies de las otras dos familias sí se encajan en $\mathbb{R}^3$, luego lo dividen en dos conjuntos abiertos (por el *Teorema de separación de JBW*). Entonces, las superficies que servirán para modelar a los objetos tridimensionales pertenecerán a alguna de estas dos

[9] El *toro* es el producto cartesiano de dos esferas, $\mathbb{T} = \mathbb{S}^2 \times \mathbb{S}^2$. De manera gráfica un toro es la parte superficial de una dona.

[10] El plano proyectivo $\mathbb{P}^2(\mathbb{R})$ es el cociente de la esfera unitaria $\mathbb{S}^2$ por la relación de equivalencia $(x_1, x_2, x_3) \sim (-x_1, -x_2, -x_3)$. En [46] se da una descripción más amplia.

familias.

Las superficies que son homeomorfas a la esfera no tienen huecos como el de una dona, mientras que las otras superficies pueden tener uno o más huecos, tantos como número de toros tenga la suma conexa equivalente. Así que este número es una invariante topológico conocido como *género*.

Hay más propiedades muy interesantes que tienen superficies cerradas. Por ejemplo, con una definición conveniente de *triángulo cerrado* se puede demostrar que es posible cubrir completamente o *teselar* a cualquier superficie cerrada con un conjunto finito de ellos, de tal forma que parezca un mosaico; a este conjunto de triángulos se le llama *triangulación* [47]. Además a través de este concepto se define la *orientación* de una superficie, sin embargo no todas tienen una orientación, pero cuando existe se dice que tal superficie es *orientable*. Gracias a esto es posible distinguir en una superficie *suave*[11] y orientable, los dos sentidos de los vectores normales a ella en cada uno de sus puntos. También se tiene que toda superficie cerrada en $\mathbb{R}^3$ (al igual que toda curva cerrada en $\mathbb{R}^2$) tiene *medida cero*[12]. Resulta que las superficies cerradas de las familias *1* y *2* son orientables, mientras que las superficies de la familia *3* no lo son.

Los conjuntos de puntos que se utilizarán para modelar a los objetos se llamarán de forma general *interfaces* y se denotarán como $\partial\Omega$, más generalmente, se le llamará interface a una $(n-1)$-variedad compacta conexa para $n > 1$ o a dos puntos para $n = 1$. Se dirá que $\partial\Omega$ tiene dimensión $n-1$. Aunque también se llamarán interfaces a la unión disjunta de dos de éstas.

Como la interface $\partial\Omega$ está encajada en $\mathbb{R}^n$ (para $n > 1$), entonces cada parte conexa lo divide en dos regiones abiertas no vacías y disjuntas, una acotada y la otra no. Al conjunto no acotado se le llamará el *exterior* de la interface, denotado como Ω^+, mientras que al otro se le llamará *interior* de la interface y se escribirá como Ω^-. Cada región es conexa por caminos. En el caso en que la interface consista de dos puntos, el intervalo abierto comprendido entre ellos será el interior, mientras que el complemento del intervalo cerrado que forman será el exterior. En ambos casos se tiene que $\partial\Omega^+ = \partial\Omega = \partial\Omega^-$. La clausura $\overline{\Omega^+}$ es equivalente a un *sólido* que tiene como frontera a la interface, $\partial(\overline{\Omega^+}) = \partial\Omega$.

La separación de los casos ($n > 1$ y $n = 1$) en la definición de interface es debido a que una 0-variedad conexa no divide al espacio $\mathbb{R}$ de tal forma que una región sea acotada y la otra no, como ocurre para variedades de dimensiones mayores a 0. Mientras tanto, la 0-variedad formada por dos puntos sí lo hace, pero no es conexa, propiedad indispensable en los otros casos. En las siguientes secciones se supondrá que la interface es una n-variedad de dimensión $n = 0, 1$ ó 2, a menos que se diga lo contrario.

[11] Cuando existe un homomorfismo diferenciable que satisface la definición de variedad.

[12] Un conjunto A tiene *medida cero* si para cualquier $\epsilon > 0$, existe un conjunto numerable $\mathcal{C}$ de rectángulos cerrados $R = [a_1, b_1] \times \cdots \times [a_n, b_n]$ para el cual $A \subset \bigcup_{R \in \mathcal{C}} R$ y $\sum_{R \in \mathcal{C}} vol(R) < \epsilon$, donde $vol(R) = (b_1 - a_1) \cdot \ldots \cdot (b_n - a_n)$. Esto quiere decir que el conjunto A es muy chico o delgado.

2.3. Representación Explícita

Ya se han caracterizado, por medio de propiedades topológicas, a los conjuntos de puntos que definen a las curvas y superficies que serán empleadas para el modelado de objetos. Ahora toca hablar sobre las formas de definir a una interface, determinando su forma y la ubicación de sus puntos en el espacio ambiente. Una de ellas es la *representación explícita*. Ésta consiste en especificar a todos y sólo aquellos puntos que estén en la interface. Puede ser nombrando a cada uno de ellos, o dando algunas propiedades que solamente sean satisfechas por estos elementos de entre todos los puntos de $\mathbb{R}^n$.

Ejemplo 2.3.1 *Aquí se dan algunos ejemplos de representaciones explícita para interfaces de dimensión* $0, 1$ *y* 2 *[49].*

1. *En* $\mathbb{R}$*:*

$$\partial\Omega = \{\ -1, 1\ \}$$

Este es un caso trivial en que la interface consiste de dos puntos solamente.

2. *En* $\mathbb{R}^2$*:*

$$\partial\Omega = \{\ \mathbf{x} \in \mathbb{R}^n :\ ||\mathbf{x}|| = 1\ \}.$$

Este es el ejemplo clásico de la circunferencia unitaria centrada en el origen.

3. *En* $\mathbb{R}^n$*:*

$$\partial\Omega = \{\ \mathbf{x} \in \mathbb{R}^n :\ ||\mathbf{x}|| = 1\ \}.$$

La interface es la esfera unitaria centrada en el origen.

Los ejemplos anteriores son casos sencillos de representar. En el primero se nombran los elementos, mientras que en los últimos dos, las interfaces se caracterizan por las normas de sus elementos, las cuales deben ser iguales a 1. Ahora bien, en el caso de la esfera se tiene que si $\mathbf{x} = (x_1, x_2, x_3)$ la norma queda como $||\mathbf{x}|| = x_1^2 + x_2^2 + x_3^2 = 1$, entonces dejando a x_3 en función de las otras variables se generan dos ecuaciones, que en conjunto describen a toda la esfera. Desde ese enfoque la interface se representa como la unión de las gráficas de dos funciones,

$$\partial\Omega = \left\{\ \mathbf{x} \in \mathbb{R}^3 :\ x_3 = \sqrt{1 - (x_1^2 + x_2^2)}\ \right\} \bigcup \left\{\ \mathbf{x} \in \mathbb{R}^3 :\ x_3 = -\sqrt{1 - (x_1^2 + x_2^2)}\ \right\}.$$

Aunque aquí no es el caso, comúnmente a esta forma de representación es a la que se le llama como representación explícita, ya que una de las variables está dada en función explícita de las restantes. Esta idea se puede generalizar en la siguiente definición.

Definición 2.3.2 *Se dice que una interface* $\partial\Omega$ *tiene una representación por medio de funciones explícitas si*

$$\partial\Omega = \bigcup_{i=1}^{m} A_i,$$

donde $A_i = \{\ \mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n :\ x_j = f_i(x_1, \ldots, \hat{x}_j, \ldots, x_n) \in f_i(D_i)\ \}$ *con* $f_i : D_i \subset \mathbb{R}^{n-1} \longrightarrow \mathbb{R}$ *función continua, cuyo dominio* D_i *es conexo por caminos, para todo* $i = 1, \ldots, m$.

La notación $\hat{x}_j$ en el vector $(x_i, \ldots, \hat{x}_j, \ldots, x_n)$, significa que la coordenada j-ésima es removida. Ahora bien, puede resultar muy difícil expresar de esta manera a una interface con forma compleja, por lo que es más útil otro tipo de representación explícita, la *representación paramétrica.*

Definición 2.3.3 *1. Una parametrización para una interface* $\partial\Omega \subset \mathbb{R}^2$ *de dimensión* 1 *es una función continua* $f : [a_1, b_1] \longrightarrow \mathbb{R}^2$, *tal que* $\partial\Omega = f([a_1, b_1])$. *A* $\partial\Omega$ *se le conoce como curva parmétrica.*

2. Para una interface $\partial\Omega \subset \mathbb{R}^3$ *de dimensión* 2 *es una función continua* $f : C \longrightarrow \mathbb{R}^3$, *tal que* $C \subset \mathbb{R}^2$ *es un conjunto conexo por caminos y compacto con* $f(C) = \partial\Omega$. *A* $\partial\Omega$ *se le llama superficie paramétrica.*

En cualquier circunstancia, al dominio de la parametrización f *se le conoce como conjunto o espacio de parámetros.*

Según la definición anterior se supone que la interface es conexa, aunque para representar interfaces disconexas, se puede particionar el dominio según el número elementos conexos y pedir que las funciones paramétricas sean "continuas por pedazos". Evidentemente para una interface de dimensión cero no es necesario una función explícita ni una parametrización para ser representada, ya que pueden ser nombrados directamente sus elementos. Por lo contrario, existen varias parametrizacones para una misma curva o superficie.

Ejemplo 2.3.4 *Se ejemplificará la definición de parametrización dando dos representaciones paramétricas diferentes para una circunferencia y dos para una esfera.*

1. La forma paramétrica de la circunferencia de radio 1 *centrada en el origen de* $\mathbb{R}^2$ *está dada por:*

$$\begin{aligned}\partial\Omega = \{\ (x_1, x_2) \in \mathbb{R}^2 : (x_1, x_2) = f(\theta) = (cos(\theta), cos(\theta)),\ \theta \in [0, 2\pi]\ \} = \\ = \{\ (x_1, x_2) \in \mathbb{R}^2 :\ (x_1, x_2) = f(\theta) = (cos(\theta), -sin(\theta)),\ \theta \in [-\pi, \pi]\ \}.\end{aligned}$$

2. Para la esfera unitaria centrada en el origen de $\mathbb{R}^3$ *se tiene:*

$$\begin{aligned}\partial\Omega = \{\ (x_1, x_2, x_3) \in \mathbb{R}^3 :\ (x_1, x_2, x_3) = f(\theta, \alpha) = (cos(\theta)cos(\alpha), sin(\theta)cos(\alpha), sin(\alpha)), \\ (\theta, \alpha) \in [0, 2\pi] \times [0, 2\pi]\ \} = \{\ (x_1, x_2, x_3) \in \mathbb{R}^3 :\ (x_1, x_2, x_3) = f(\theta, \alpha) = \\ = (cos(\theta)cos(2\alpha), sin(\theta)cos(2\alpha), -sin(2\alpha)),\ (\theta, \alpha) \in [0, 2\pi] \times [-\pi/2, \pi/2]\ \}.\end{aligned}$$

No hay mucha diferencia entre las parametrizaciones de las interfaces anteriores, pero son esencialmente diferentes. Sin embargo, lo realmente importante es que describan a la curva o superficie de interés.

Se pueden hacer analogías con la forma en cómo actúan las parametrizaciones con su dominio. Por ejemplo, en el caso de la circunferencia unitaria, sus parametrizaciones toman el segmento de recta $[0, 2\pi]$ y lo pegan por sus extremos deformándolo hasta obtener la circunferencia, como si tal segmento fuera de plastilina que puede deformarse sin romper ni pegar, solo en los extremos; lo mismo ocurre con la esfera, las parametrizaciones hacen que el rectángulo de su dominio parezca de un material flexible que puede estirarse y encogerse sin romperse ni pegarse, a excepción de sus bordes, de tal forma que lo deforman hasta convertirlo en una esfera.

Hasta este punto solamente se ha hablado sobre el tipo de representación explícita, con la cual no es posible especificar directamente ni el interior ni el exterior de la interface. Pero es de saber que cuando sus parametrizaciones son hasta cierto punto " suaves ", se puede extraer a partir de ellas la dirección normal por medio del gradiente, así como la curvatura media de la interface en un punto específico, etc. Todo esto también lo proporciona una representación implícita, que además permite la identificación directa tanto del interior como del exterior, ya que valoriza a todo el espacio de tal manera que estas dos regiones son claramente diferenciadas.

2.4. Representación Implícita

En esta sección se introduce la representación implícita de una interface por medio de una función que a cada punto del espacio $\mathbb{R}^n$ le asigna un valor real único. Estas funciones definen a las interfaces por medio de sus conjuntos de nivel, es decir, cada interface es el conjunto donde una función escalar toma un valor constante, como se observa en el siguiente apartado.

2.4.1. Funciones Implícitas

Definición 2.4.1 *En general, una función implícita de un conjunto $A \subset \mathbb{R}^n$, es una función escalar $\phi : \mathbb{R}^n \longrightarrow \mathbb{R}$, tal que*

$$A = \{\ \mathbf{x} \in \mathbb{R}^n : \phi(\mathbf{x}) = a\ \},$$

con $a \in \mathbb{R}$ fijo. Al conjunto de la derecha de la expresión anterior se le conoce como conjunto de nivel o isocontorno de nivel a. Pero para que represente a una interface $A = \partial\Omega \subset \mathbb{R}^n$, adicionalmente se pide que la función ϕ sea continua y satisfaga

$$\Omega^- = \{\ \mathbf{x} \in \mathbb{R}^n : \phi(\mathbf{x}) < a\ \}\ y$$

$$\Omega^+ = \{\ \mathbf{x} \in \mathbb{R}^n : \phi(\mathbf{x}) < a\ \}.$$

De esta forma si $\partial\Omega$ es una curva se le llama curva implícita (o isocurva) y si es una superficie se le llama superficie implícita (o isosuperficie).

La definición anterior dice que la interface puede ser vista como la intersección de la gráfica de ϕ con un conjunto $\mathbb{R}^{n-1}$. Notar que los valores de la función implícita en el interior son menores que un umbral dado y en el exterior son mayores que ese umbral, pero hubiera sido equivalente haber definido a ambos conjuntos con las desigualdades invertidas, puesto que no importa la función implícita en sí misma, sino que determine tanto a la interface, al interior y al exterior al mismo tiempo con la comparación de sus valores con respecto a un número.

Observación 2.4.2 *Cuando una interface está representada por un isocontorno $\phi = a$, se puede considerar otra función escalar continua $\hat{\phi} = \phi - a$, cuyo isocontorno $\hat{\phi} = 0$ describe a la misma interface. Al igual que los valores de ϕ en el interior son negativos, mientras que en el exterior son positivos. Con esto la función ϕ es una función que representa implícitamente a la interface. Además comparten otras propiedades. La existencia de las derivadas parciales de ϕ implica la existencia de las parciales de $\hat{\phi}$, y recíprocamente, que también son iguales ya que la suma de una constante se convierte en cero bajo derivación. Debido a esta igualdad también cuentan con las mismas propiedades que dependen de sus derivadas parciales.*

Por la observación anterior se puede restringir la representación implícita de interfaces a funciones implícitas que describan a éstas con sus isocontornos de nivel cero. Al igual que en las parametrizaciones existen varias funciones escalares que representan implícitamente, e inclusive con el mismo isocontorno, a una interface; una puede ser diferenciable en un punto mientras que otra no necesariamente. Se darán unos ejemplos en esta sección para ciertas interfaces y más adelante se representarán con otras funciones implícitas llamadas *funciones distancia con signo*.

Ejemplo 2.4.3 *Para ejemplificar se toman las interfaces del **Ejemplo 2.3.1.**, dando las funciones implícitas que las describen, así mismo se expresan el interior y el exterior correspondientes a cada una de ellas.*

1. *En $\mathbb{R}$, una función implícita que representa a la interface que consta de los puntos -1 y 1 es (ver Figura 2.3, parte superior):*

$$\phi(x) = x^2 - 1.$$

   El interior es el intervalo abierto de radio 1 centrado en cero $\Omega^- = \{\ x \in \mathbb{R} :\ |x| < 1\ \} = (-1, 1)$. El exterior es $\Omega^+ = \{\ x \in \mathbb{R} : |x| > 1\ \} = (-\infty, -1) \cup (1, \infty)$.

2. *En $\mathbb{R}^2$, para representar a la circunferencia unitaria con centro en el origen se uitliza la función implícita (ver Figura 2.3, parte inferior):*

$$\phi(\mathbf{x}) = x_1^2 + x_2^2 - 1,$$

donde $\mathbf{x} = (x_1, x_2)$. *El interior de la interface es el disco abierto con radio* 1, $\Omega^- = \{\ \mathbf{x} \in \mathbb{R}^2 : x_1^2 + x_2^2 < 1\ \} = \mathbb{B}_1(\mathbf{0})$. *Su complemento es el disco cerrado con radio* 1 *centrado en el origen* $\Omega^+ = \{\ \mathbf{x} \in \mathbb{R}^2 :\ x_1^2 + x_2^2 > 1\ \} = \overline{\mathbb{B}_1(\mathbf{0})}^c$.

3. *En* $\mathbb{R}^3$, *la esfera unitaria es posible representarla con la siguiente función implícita:*

$$\phi(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 1,$$

tal que $\mathbf{x} = (x_1, x_2, x_3)$. *El interior es la bola abierta de radio* 1 *centrado en el origen,* $\Omega^- = \{\ \mathbf{x} \in \mathbb{R}^3 :\ x_1^2 + x_2^2 + x_3^2 < 1\ \} = \mathbb{B}_1(\mathbf{0})$. *Su exterior es el complemento de la bola cerrada de radio 1 centrada en el origen,* $\Omega^+ = \{\ \mathbf{x} \in \mathbb{R}^3 :\ x_1^2 + x_2^2 + x_3^2 > 1\ \} = \overline{\mathbb{B}(\mathbf{0})}^c$.

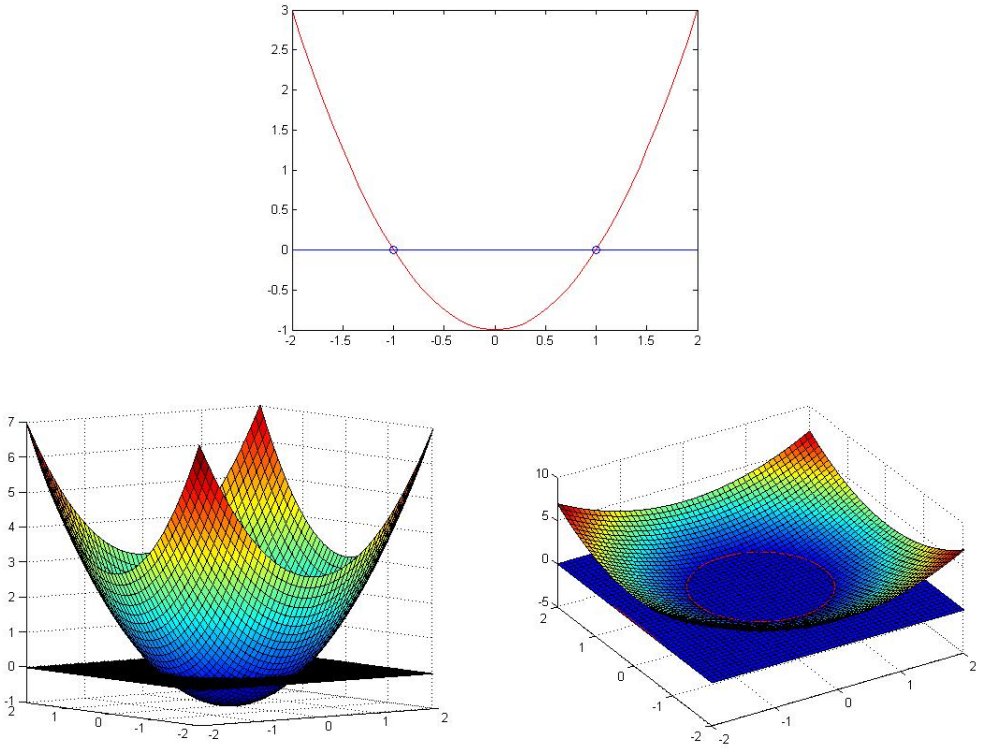


Figura 2.3: (*superior*) Representación implícita de $\{-1, 1\}$. (*inferior*) Representación implícita del círculo unitario centrado en el origen.

El término interface no sólo incluye la modelación de un único sólido, sino también de varios. E incluso se pueden realizar operaciones booleanas (operaciones de conjuntos) entre los modelos sólidos de estos objetos. En general, las operaciones booleanas de modelos sólidos no son cerradas, por ejemplo, la intersección de dos de ellos puede dar como resultado modelos no sólidos como líneas o puntos. Para arreglar esto se redefinen las operaciones booleanas resultando en un conjunto de operaciones cerradas

denominadas *operaciones booleanas regularizadas* [11]. Sin embargo, es de gran ayuda tener un conjunto de operaciones boolenas, no regularizadas, entre sólidos por medio de las funciones implícitas que los representan y sin la necesidad de crear otra función totalmente independiente, como ocurre con la representación paramétrica.

Proposición 2.4.4 *Dadas dos funciones implícitas* $\phi_1 : \mathbb{R}^n \longrightarrow \mathbb{R}$ *y* $\phi_2 : \mathbb{R}^n \longrightarrow \mathbb{R}$*, supongamos que* Ω_1^- *y* Ω_2^- *son los correspondientes interiores de las interfaces que representan, entonces se cumplen las siguientes afirmaciones:*

1. $\phi = \text{mín}\{\ \phi_1, \phi_2\ \}$ *es una función continua tal que*

$$\partial(\Omega_1^- \cup \Omega_2^-) = \{\ \mathbf{x} \in \mathbb{R}^n :\ \phi(\mathbf{x}) = 0\ \}\ y\ \Omega_1^- \cup \Omega_2^- = \{\ \mathbf{x} \in \mathbb{R}^n :\ \phi(\mathbf{x}) < 0\ \}.$$

2. $\phi = \text{máx}\{\ \phi_1, \phi_2\ \}$ *es una función continua y cumple con*

$$\partial(\overline{\Omega_1^-} \cap \overline{\Omega_2^-}) = \{\ \mathbf{x} \in \mathbb{R}^n :\ \phi(\mathbf{x}) = 0\ \}\ y\ \Omega_1^- \cap \Omega_2^- = \{\ \mathbf{x} \in \mathbb{R}^n :\ \phi(\mathbf{x}) < 0\ \}.$$

3. $\phi = \text{máx}\{\ \phi_1, -\phi_2\ \}$ *es una función continua tal que*

$$\partial(\overline{\Omega_1^-} \backslash \Omega_2^-) = \{\ \mathbf{x} \in \mathbb{R}^n :\ \phi(\mathbf{x}) = 0\ \}\ y\ \Omega_1^- \backslash \overline{\Omega_2^-} = \{\ \mathbf{x} \in \mathbb{R}^n :\ \phi(\mathbf{x}) < 0\ \},$$

   donde el símbolo $\backslash$ *significa diferencia de conjuntos.*

4. $\phi = -\phi_1$ *es una función continua con*

$$\partial\Omega_1^- = \{\ \mathbf{x} \in \mathbb{R}^n :\ \phi(\mathbf{x}) = 0\ \}\ y\ \overline{\Omega_1^-}^c = \{\ \mathbf{x} \in \mathbb{R}^n :\ \phi(\mathbf{x}) < 0\ \}.$$

Las operaciones de conjuntos que se llevan a cabo de forma implícita con estas funciones son, en el orden en el que aparecen en la proposición, la unión, la intersección, la diferencia y el complemento de los sólidos generados por las interfaces iniciales.

2.4.2. Estructuras definidas a partir de la Representación Implícitas

Se pueden definir algunas estructuras relacionadas a las interfaces, a partir de las funciones implícitas cuando existen las derivadas parciales, como el *gradiente* y la *curvatura media*.

Definición 2.4.5 *Si en un punto* $\mathbf{x} \in \mathbb{R}^n$ *existe cada derivada parcial en la dirección* x_i*,* $\frac{\partial \phi}{\partial x_i}$ *, el gradiente de la función implícita en ese punto está definido por*

$$\nabla\phi = \left(\frac{\partial \phi}{\partial x_1}, \ldots, \frac{\partial \phi}{\partial x_n}\right). \tag{2.5}$$

El gradiente es normal a los conjuntos de nivel de ϕ, en particular, en cualquier punto donde sea suave la interface y que el gradiente no se anule éste apunta en la misma dirección que el *vector normal unitario* $\mathbf{N}$ a la interface, el cual apunta hacia el exterior desde dichos puntos. Este vector puede ser definido a partir del gradiente.

Definición 2.4.6 *Suponiendo que existe el gradiente $\nabla\phi$ y no se anula en un punto $\mathbf{x}$, se define el vector normal unitario exterior en ese punto como*

$$\mathbf{N} = \frac{\nabla\phi}{|\nabla\phi|}. \tag{2.6}$$

Notar que la normal unitaria no se limita a ser calculada solamente en los puntos de la interface sino en cualquier punto del dominio en donde exista el gradiente de ϕ, excepto en sus *puntos críticos*, puntos donde el gradiente es cero. Cuando existe la normal unitaria, junto con ciertas condiciones extras, es posible hallar la *curvatura media* de la interface en un punto determinado.

Definición 2.4.7 *Sea $\mathbf{N} = (N_1, \ldots, N_n)$ con $N_i = \frac{\partial\phi}{\partial x_i} = \phi_{x_i}$. Si existen las parciales dobles y mixtas de ϕ en un punto $\mathbf{x} \in \partial\Omega$, la curvatura media de la interface en ese punto está definida como la divergencia de la normal $\vec{N}$, es decir,*

$$\kappa = \nabla \cdot \mathbf{N} = \frac{\partial N_1}{\partial x_1} + \cdots + \frac{\partial N_n}{\partial x_n}, \tag{2.7}$$

donde $\nabla = (\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n})$ es el operador nabla.

Observación 2.4.8 *Si se denotan las parciales dobles de ϕ como $\frac{\partial^2\phi}{\partial x_i^2} = \phi_{x_i x_i}$ y las parciales mixtas como $\frac{\partial^2\phi}{\partial x_i \partial x_j} = \phi_{x_i x_j}$, entonces la curvatura media queda determinada por las parciales de $\phi : \mathbb{R}^n \longrightarrow \mathbb{R}$ de la siguiente manera:*

$$\kappa = \nabla \cdot \left(\frac{\nabla\phi}{||\nabla\phi||}\right) = \begin{cases} 0, & si\ n = 1, \\ (\phi_x^2\phi_{yy} - \phi_x\phi_y\phi_{xy} - \phi_x\phi_y\phi_{yx} + \phi_y^2\phi_{xx})/||\nabla\phi||^3, & si\ n = 2, \\ (\phi_x^2\phi_{yy} - \phi_x\phi_y\phi_{xy} - \phi_x\phi_y\phi_{yx} + \phi_y^2\phi_{xx} + \phi_x^2\phi_{zz} - & \\ -\phi_x\phi_z\phi_{xz} - \phi_x\phi_z\phi_{zx} + \phi_z^2\phi_{xx} + \phi_y^2\phi_{zz} - \phi_y\phi_z\phi_{yz} - & si\ n = 3. \\ -\phi_y\phi_z\phi_{zy} + \phi_z^2\phi_{yy})/||\nabla\phi||^3 & \end{cases} \tag{2.8}$$

con $x_1 = x$, $x_2 = y$ y $x_3 = z$.

En este apartado sólo se habla sobre la curvatura media de las interfaces, pero en la referencia [40] hay fórmulas para encontrar la curvatura gaussiana de una superficie implícita.

A cualquier conjunto de $\mathbb{R}^n$ se le puede asignar una función que indica si un punto de este espacio se encuentra en el conjunto, dando en ese caso el valor 1, o si está en su complemento, cuyo valor asignado sería 0. A esta función se le conoce como *función indicadora* o *función característica*. Pero en el caso especial en que una interface sea representada por una función implícita ϕ, la función característica χ^- del conjunto $\overline{\Omega^-}$, queda como:

$$\chi^-(\mathbf{x}) = \begin{cases} 1, & si\ \phi(\mathbf{x}) \leqslant 0, \\ 0, & si\ \phi(\mathbf{x}) > 0. \end{cases} \tag{2.9}$$

Mientras que la función característica χ^+ del exterior queda determinada por:

$$\chi^+(\mathbf{x}) = \begin{cases} 0, & si\ \phi(\mathbf{x}) \leqslant 0, \\ 1, & si\ \phi(\mathbf{x}) > 0. \end{cases} \tag{2.10}$$

Si una una función escalar f está definida y es integrable sobre todo el espacio $\mathbb{R}^n$ se puede calcular la integral sobre el interior de la interface con la función característica de $\overline{\Omega^-}$,

$$\int_{\Omega^-} f(\mathbf{x})d\mathbf{x} = \int_{\mathbb{R}^n} f(\mathbf{x})\chi^-(\mathbf{x})d\mathbf{x}. \tag{2.11}$$

Esta integral conocida como *integral volumen* cuando la interface es una superficie cerrada, *integral de área* si la interface es una curva cerrada e *integral de línea* si es un conjunto de dos puntos. La inclusión de la interface en la función característica χ^- no afecta al resultado de la integral en el interior, ya que tiene medida cero; así que pudo haberse omitido o incluido al exterior para la definición de χ^+. Cuando la función f es la constante 1, el resultado de la integral sobre Ω^- es lo que se puede considerar el *volumen* de la interface, siendo el volumen propiamente dicho del interior de una superficie cerrada en el epacio, el área interior para una curva cerrada en el plano y la longitud del segmento localizado entre dos puntos de la recta real. En muchas aplicaciones en lugar de utilizar las funciones características tal cual son puestas en términos de la *función de Heaviside*:

$$H(\phi) = \begin{cases} 0, & si\ \phi \leqslant 0, \\ 1, & si\ \phi > 0. \end{cases} \tag{2.12}$$

De esta forma las funciones características se ven como $\chi^+ = H(\phi)$ y $\chi^- = 1 - H(\phi)$, por tanto la ecuación (2.11) también cambia

$$\int_{\Omega^-} f(\mathbf{x})d\mathbf{x} = \int_{\mathbb{R}^n} f(\mathbf{x})(1 - H(\phi))d\mathbf{x}. \tag{2.13}$$

2.5. Representación Implícta a través de Funciones Distancia con Signo

En esta sección se introduce un tipo especial de función implícita, llamada *función distancia con signo* o *distancia orientada*. Estas funciones tienen varias propiedades, además simplifican muchas expresiones y ecuaciones, así como permiten hacer más precisos y estables ciertos cálculos con respecto a la evolución de forma de las interfaces que representan. Estas funciones se pueden definir para representar conjuntos más generales que curvas y superficies cerradas, por lo que se presentará la teoría de la forma más general posible, para deducir como caso particular los casos de interés. Se definirán algunos conceptos relacionados y se darán a conocer las propiedades más importantes de este tipo de funciones. Pero primero se estudian las *funciones distancia* y se considera al espacio $\mathbb{R}^n$ con la topología usual.

2.5.1. Funciones distancia

Definición 2.5.1 *Una función distancia con respecto a un conjunto $\varnothing \neq A \subset \mathbb{R}^n$ es una función escalar $d_A : \mathbb{R}^n \longrightarrow \mathbb{R}$ definida como*

$$d_A(\mathbf{x}) = \text{ínf}\{\ ||\mathbf{x} - \mathbf{y}|| :\ \mathbf{y} \in A\ \}. \tag{2.14}$$

El símbolo ínf hace referencia al *ínfimo* del conjunto de todas las distancias que hay entre un punto fijo cualquiera del espacio $\mathbb{R}^n$ y los puntos de un conjunto arbitrario A. Por tanto, la función distancia mide cuan cerca del conjunto A se encuentra dicho punto. En un sentido, esta función generaliza el concepto de norma en el espacio euclídeo $\mathbb{R}^n$, que mide la distancia de cualquier punto con respecto al origen. Esta función está bien definida, ya que si se tiene en cuenta que la distancia entre dos puntos siempre es mayor o igual a cero entonces siempre está asegurada la existencia del ínfimo de cualquier conjunto de distancias. Enseguida se dan algunas propiedades importantes.

Proposición 2.5.2 *Sean A y B dos subconjuntos no vacíos de $\mathbb{R}^n$, supongamos que d_A y d_B son las correspondientes funciones distancia, entonces:*

1. *d_A es una función que cumple la condición de Lipschitz, esto es, para todo $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ se cumple*

$$|d_A(\mathbf{x}) - d_A(\mathbf{y})| \leqslant ||\mathbf{x} - \mathbf{y}||. \tag{2.15}$$

2. *d_A es una función uniformemente continua[13] en todo $\mathbb{R}^n$.*

3. *Para cualquier $\mathbf{x} \in \mathbb{R}^n$, existe $\mathbf{y}_c \in \overline{A}$ tal que $d_A(\mathbf{x}) = ||\mathbf{x} - \mathbf{y}_c||$, además $d_A = d_{\overline{A}}$.*

4. *$\overline{A} = \{\mathbf{x} \in \mathbb{R}^n :\ d_A(\mathbf{x}) = 0\ \}$.*

5. *Si $\overline{A} \subset \overline{B}$, entonces $d_A \geqslant d_B$.*

6. *d_A es (Fréchet[14]) diferenciable casi donde sea[15] en $\mathbb{R}^n$.*

7. *d_A satisface*

$$||\nabla d_A|| \leqslant 1. \tag{2.16}$$

La mayoría de las demostraciones pueden encontrarse en [48], otras son resultados básicos de cálculo de varias variables. Sin embargo se puede demostrar casi de inmediato que *1* implica *2*, mientras que *2* implica la continuidad de d_A en todo el espacio $\mathbb{R}^n$. Como consecuencia de *3* se tiene que la clausura

[13] Se dice que una función $f : D \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ es *uniformemente continua* si para todo $\epsilon > 0$ existe $\delta > 0$, tal que para $\mathbf{x}, \mathbf{y} \in D$ siempre que se cumpla $||\mathbf{x} - \mathbf{y}|| < \delta$, se tiene $|f(\mathbf{x}) - f(\mathbf{y})| < \epsilon$.

[14] Una función $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ es *Fréchet diferenciable* en $\mathbf{x} \in \mathbb{R}^n$ si existe el límite

$$\lim_{h \to 0} \frac{f(\mathbf{y}) - f(\mathbf{x}) - \nabla f(\mathbf{x}) \cdot (\mathbf{x} - \mathbf{y})}{||\mathbf{x} - \mathbf{y}||} = 0$$

[15] Se dice que una propiedad se cumple casi en todos lados, casi donde sea o casi en todas partes en $\mathbb{R}^n$, si tal propiedad solamente no se cumple en un conjunto de medida cero.

del conjunto es igual al isocontorno de nivel cero de la función distancia. Al ser el conjunto A cerrado la propiedad *4* dice que el ínfimo se puede cambiar por mínimo, es decir, para cualquier punto en el espacio ambiente siempre existe un punto del conjunto que es el más cercano a él, no necesariamente es único. Si un conjunto está contenido en otro, entonces cada punto que no está en ninguno de los dos, se encuentran más lejos del conjunto que está contenido que el que lo contiene, como señala *5*. Para demostrar *6* se aplica *1* junto con el *Teorema de Rademacher*[16]; esta propiedad afirma que la función distancia casi siempre es diferenciable, excepto en un conjunto muy chico, casi "sin importancia". Por último, según *7*, siempre que exista ∇d_A, su norma no será mayor a 1. Se tiene un rango acotado para el tamaño de los vectores del gradiente.

Hay algunos conjuntos importantes que se definirán a continuación como el *conjunto de proyecciones* de un punto y los *esqueletos*. Éstos tienen una relación con la diferenciación de la función distancia como se verá más adelante.

Definición 2.5.3 *Dado $\varnothing \neq A \subset \mathbb{R}^n$, el conjunto de proyecciones de $\mathbf{x}$ sobre A está dado por:*

$$\Pi_A(\mathbf{x}) = \{\ \mathbf{p} \in \overline{A} :\ d_A(\mathbf{x}) = ||\mathbf{x} - \mathbf{p}||\ \}. \tag{2.17}$$

Los elementos de $\Pi_A(\mathbf{x})$ son llamados las proyecciones de $\mathbf{x}$ sobre A.

Observación 2.5.4 *El conjunto de proyecciones consiste de los puntos más cercanos a la clausura del conjunto en cuestión. Esto implica que para $A \neq \varnothing$, si $\mathbf{x} \in \overline{A}$, entonces $\Pi_A(\mathbf{x})$ tiene un sólo elemento, que es el mismo $\mathbf{x}$. Luego, cuando $A^c \neq \varnothing$, $\Pi_{A^c}(\mathbf{x}) = \{\ \mathbf{x}\ \}$ para cualquier punto $\mathbf{x} \in \overline{A^c}$.*

Definición 2.5.5 *Si $A \subset \mathbb{R}^n$ no es vacío, el esqueleto exterior de A es definido como:*

$$\mathrm{Sk}_{ext}(A) = \{\ \mathbf{x} \in \mathbb{R}^n :\ \Pi_A(\mathbf{x})\ \textit{tiene más de un elemento}\ \}. \tag{2.18}$$

Si además $A^c \neq \varnothing$, el esqueleto interior de A se define de manera similar (ver Figura 2.4),

$$\mathrm{Sk}_{int}(A) = \{\ \mathbf{x} \in \mathbb{R}^n :\ \Pi_{A^c}(\mathbf{x})\ \textit{tiene más de un elemento}\ \}. \tag{2.19}$$

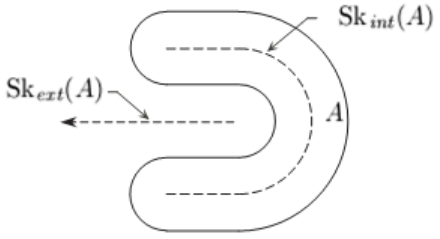


Figura 2.4: Esqueleto interior y exterior del conjunto A.

[16]El teorema de Radamacher dice lo siguiente: Si U es un subconjunto abierto de $\mathbb{R}^n$ y $f : U \longrightarrow \mathbb{R}^m$ es una función Lipschitz, entonces f es una función Fréchet diferenciable casi en todas partes en U.

Observación 2.5.6 *De la* **Observación 1.6.4** *y las definiciones anteriores se puede concluir:*

$$\mathrm{Sk}_{ext}(A) \subset \mathrm{int}A^c = \overline{A}^c \text{ y } \mathrm{Sk}_{int}(A) \subset \mathrm{int}A = \overline{A^c}^c.$$

Se quiere caracterizar, o al menos conocer cómo es la mayor parte del conjunto de puntos en donde no existe el gradiente de d_A, conocido como el *conjunto de singularidades del gradiente de* d_A, el cual se denota $\mathrm{Sing}(\nabla d_A)$. Para esto se procederá hacer una relación entre la función distancia d_A con la diferenciabilidad de su cuadrado d_A^2.

Teorema 2.5.7 *Sea* $\varnothing \neq A \subset \mathbb{R}^n$ *y* $\mathbf{x} \in \mathbb{R}^n$*, entonces:*

1. *El conjunto* $\Pi_A(\mathbf{x})$ *es no vacío, compacto y para cualquier* $\mathbf{x} \notin \overline{A}$*, se tiene que*

$$\Pi_A(\mathbf{x}) \subset \partial\overline{A}.$$

2. d_A^2 *es (Fréchet) diferenciable en* $\mathbf{x}$ *si y sólo si el conjunto* $\Pi_A(\mathbf{x})$ *contiene un único elemento, denotado como* $p_A(\mathbf{x})$*. Además se da la igualdad*

$$p_A(\mathbf{x}) = \mathbf{x} - \frac{1}{2}\nabla d_A^2(\mathbf{x}). \tag{2.20}$$

Observación 2.5.8 *De 2 se puede concluir que*

$$\mathrm{Sk}_{ext}(A) = \{\mathbf{x} \in \mathbb{R}^n : \nexists \nabla d_A^2(\mathbf{x})\} \subset \mathrm{int}A^c = \overline{A}^c,$$

y como la existencia de $\nabla d_A(\mathbf{x})$ *implica la existencia de* $\nabla d_A^2(\mathbf{x})$*, ocurre que*

$$\mathrm{Sing}(\nabla d_A) = \mathrm{Sk}_{ext}(A) \cup \mathrm{C}_{ext}(A),$$

donde

$$\mathrm{C}_{ext}(A) = \{\ \mathbf{x} \in \mathbb{R}^n : \exists \nabla d_A^2(\mathbf{x}) \text{ y } \nexists \nabla d_A(\mathbf{x})\}$$

es llamado la quebradura exterior de A. Paralelamente, si $A^c \neq \varnothing$*,* d_{A^c} *cumple los dos enunciados de la proposición anterior, definiendo por tanto la quebradura exterior de A como*

$$\mathrm{C}_{int}(A) = \{\ \mathbf{x} \in \mathbb{R}^n : \exists \nabla d_{A^c}^2(\mathbf{x}) \text{ y } \nexists \nabla d_{A^c}(\mathbf{x})\}.$$

Observación 2.5.9 *Es de notar las relaciones que hay entre las definiciones de los conjuntos introducidos en esta sección y teniendo en cuenta que* $d_A = d_{\overline{A}}$*:*

$$\mathrm{Sk}_{ext}(A) = \mathrm{Sk}_{ext}(\overline{A}) = \mathrm{Sk}_{int}(A^c),\ \mathrm{Sk}_{int}(A) = \mathrm{Sk}_{int}(\overline{A}) = \mathrm{Sk}_{ext}(A^c),\ \mathrm{C}_{ext}(A) = \mathrm{C}_{ext}(\overline{A}) = \mathrm{C}_{int}(A^c),$$

$$\mathrm{C}_{int}(A) = \mathrm{C}_{int}(\overline{A}) = \mathrm{C}_{ext}(A^c) \text{ y } \mathrm{Sing}(\nabla d_A) = \mathrm{Sing}(\nabla d_{\overline{A}}).$$

Enseguida se dará una lista de propiedades que darán mucha información acerca de las funciones distancia, entre ello permitirán calcular directamente el gradiente en casi cualquier parte. Los detalles de la demostración se dan en [48].

Teorema 2.5.10 *1. Sea* $\varnothing \neq A \subset \mathbb{R}^n$. *Si* $\nabla d_A(\mathbf{x})$ *existe en un punto* $\mathbf{x} \in \mathbf{R}^n$, *entonces* $\Pi_A(\mathbf{x}) = \{\ p_A(\mathbf{x})\ \}$ *y*

$$d_A(\mathbf{x}) = ||p_A(\mathbf{x}) - \mathbf{x}|| \text{ y } \nabla d_A(\mathbf{x}) = \begin{cases} 0, & si\ \mathbf{x} \in \overline{A}, \\ \frac{\mathbf{x}-p_A(\mathbf{x})}{||\mathbf{x}-p_A(\mathbf{x})||}, & si\ \mathbf{x} \notin \overline{A}. \end{cases}$$

2. Para $\varnothing \neq A \subset \mathbb{R}^n$ *y* $\mathbf{x} \in \mathbb{R}^n \backslash \partial A$, d_A *es diferenciable en* $\mathbf{x}$ *si y solamente si* d_A^2 *es diferenciable en* $\mathbf{x}$.

3. Si $\varnothing \neq A \subset \mathbb{R}^n$, *entonces* $\mathrm{Sing}(\nabla d_A)$ *tiene medida cero.*

4. Para $\mathbf{x} \in \mathbb{R}^n \backslash \mathrm{Sing}(\nabla d_A)$, *es decir, para casi todo* $\mathbf{x}$ *se tiene que*

$$\chi_{\overline{A}}(\mathbf{x}) = 1 - ||\nabla d_A(\mathbf{x})||\ y$$

$$\chi_{\mathrm{int}A^c}(\mathbf{x}) = ||\nabla d_A(\mathbf{x})||,$$

tal que χ_A *es la función indicadora del conjunto* A.

5. Dado $\mathbf{x} \in \mathbb{R}^n$, $\alpha \in [0,1]$, $\mathbf{p} \in \Pi_A(\mathbf{x})$ *y* $\mathbf{x}_\alpha = \mathbf{p} + \alpha(\mathbf{x} - \mathbf{p})$,

$$d_A(\mathbf{x}_\alpha) = ||\mathbf{x}_\alpha - \mathbf{p}|| = \alpha||\mathbf{x} - \mathbf{p}|| = \alpha d_A(\mathbf{x}),$$

además para cualquier $\alpha \in [0,1]$, $\Pi_A(\mathbf{x}_\alpha) \subset \Pi_A(\mathbf{x})$. *En particular, cuando* $\Pi_A(\mathbf{x})$ *tiene un único elemento,* $\Pi_A(\mathbf{x}_\alpha) = \{\ p_A(\mathbf{x})\ \}$ *y* $\nabla d_A^2(\mathbf{x}_\alpha)$ *existe para toda* $\alpha \in [0,1]$. *Si de forma adicional* $\mathbf{x} \notin \overline{A}$, $\nabla d_A(\mathbf{x}_\alpha)$ *existe para* $\alpha \in (0,1]$ *y* $\nabla d_A(\mathbf{x}_\alpha) = \nabla d_A(\mathbf{x})$.

En la primera parte del teorema se da una fórmula explícita para hallar el gradiente de la función distancia, siempre y cuando éste exista. La segunda parte garantiza la existencia de tal gradiente en cualquier punto fuera de la frontera del conjunto A sabiendo simplemente que tiene un único punto más cercano a la clausura $\overline{A}$. En primera instancia no se sabe nada del gradiente en la frontera de A, sin embargo, no hay que preocuparse tanto, ya que *3* dice que el conjunto de puntos donde el gradiente de la función distancia no existe, es pequeño. Como consecuencia de *2* y *3*, $\mathrm{C}_{ext}(A) \cup \mathrm{C}_{int}(A) \subset \partial A$, y cada una de las quebraduras así como los esqueletos tienen medida cero. Para los puntos en donde el gradiente exista, las funciones indicadoras de $\overline{A}$ y de $\mathrm{int}A^c$, se pueden expresar en términos de él. El último apartado tiene sentido ya que el segmento de línea que une a un punto con su punto más cercano a $\overline{A}$, es el camino más corto que hay a la clausura de A desde él, entonces cualquier elemento de ese segmento efectivamente debe tener el mismo punto más cercano a $\overline{A}$. Además, si existe $\nabla d_A(\mathbf{x})$ y no es cero, para cualquier punto $\mathbf{x}_\alpha$ sobre el segmento de recta, excepto para $p_A(\mathbf{x}) \in \overline{A}$, $\nabla d_A(\mathbf{x}_\alpha)$ existe, no es nulo y su negativo apunta en la dirección de descenso más rápido, en otras palabras, es un vector en $\mathbf{x}$ que apunta hacia $p_A(\mathbf{x})$.

Se han visto algunas propiedades sobre las funciones distancia, muchas de las cuales se heredan a las funciones distancia con signo, que serán de utilidad conocer más adelante en el trabajo.

2.5.2. Funciones Distancia con Signo

Este tipo de función proporciona un punto de vista de conjunto de nivel para el conjunto A, en donde el nivel cero de la función distancia con signo es la frontera de A.

Definición 2.5.11 *Sea $A \subset \mathbb{R}^n$ con $\partial A \neq \varnothing$. La función distancia con signo o función distancia orientada con respecto a A, es una función escalar $b_A : \mathbb{R}^n \longrightarrow \mathbb{R}$ definida como*

$$b_A(\mathbf{x}) = d_A(\mathbf{x}) - d_{A^c}(\mathbf{x}) \tag{2.21}$$

donde d_A y d_{A^c} son las funciones distancia con respecto a A y A^c, respectivamente

La restricción $\partial A \neq \varnothing$ al principio de la definición es debido a que esto es equivalente a pedir al mismo tiempo que A y A^c sean no vacíos, lo cual es imprescindible para que las respectivas funciones distancia estén definidas.
Una forma equivalente de ver a una función distancia con signo con respecto a un conjunto, en términos de su frontera es

$$b_A(\mathbf{x}) = \begin{cases} d_A = d_{\partial A}, & si\ \mathbf{x} \in \text{int}A^c, \\ 0, & si\ \mathbf{x} \in \partial A, \\ -d_{A^c} = -d_{\partial A}, & si\ \mathbf{x} \in \text{int}A. \end{cases} \tag{2.22}$$

Observación 2.5.12 *Hay que notar que $b_{A^c} = -b_A$, tal que en el interior de A el valor de b_A es negativo, mientras que en el interior de A^c (o exterior de A) es positivo. También se puede deducir de la forma anterior que*

$$\Pi_{\partial A}(\mathbf{x}) = \begin{cases} \Pi_A(\mathbf{x}), & si\ \mathbf{x} \in \text{int}A^c, \\ \{\ \mathbf{x}\ \}, & si\ \mathbf{x} \in \partial A, \\ \Pi_{A^c}(\mathbf{x}), & si\ \mathbf{x} \in \text{int}A. \end{cases} \tag{2.23}$$

Por tanto $\text{Sk}_{int}(A) = \varnothing$ *y*

$$\text{Sk}_{ext}(\partial A) = \text{Sk}_{int}(A) \cup \text{Sk}_{ext}(A). \tag{2.24}$$

Al igual que para las funciones distancia, existen conjuntos análogos a los *esqueletos*, a las *quebraduras* y al *conjunto de singularidades del gradiente* para las funciones distancia con signo. Sólo que en este caso se unifican en un solo conjunto las versiones interiores y exteriores que surgían anteriormente, ya que se involucran las funciones distancia de d_A y de d_{A^c}.

Definición 2.5.13 *El esqueleto de un conjunto A con frontera no vacía, está definido por:*

$$\text{Sk}(A) = \text{Sk}_{ext}(\partial A).$$

El conjunto de singularidades para el gradiente ∇b_A es:

$$\text{Sing}(\nabla b_A) = \{\ \mathbf{x} \in \mathbb{R}^n :\ \nexists \nabla b_A(\mathbf{x})\ \}.$$

La quebradura para b_A está definida como:

$$C_b(A) = \{\ \mathbf{x} \in \mathbb{R}^n :\ \exists \nabla b_A^2(\mathbf{x})\ y\ \nexists \nabla b_A(\mathbf{x})\ \}.$$

Ahora bien, muchas consecuencias de la definición de función distancia con signo se dan debido a que esta función consiste de una resta de funciones distancia; para detalles de la demostración se puede consultar [48].

Teorema 2.5.14 *Sean A y B dos conjuntos con frontera no vacía, si b_A y b_B son las respectivas funciones distancia con signo, entonces:*

1. *Se cumplen las siguientes implicaciones y equivalencias:*

   a) $B \subset A \Longrightarrow b_A \leqslant b_B$.

   b) $A = B \Longrightarrow b_A = b_B$.

   c) $b_A \leqslant b_B \Longleftrightarrow \overline{B} \subset \overline{A}\ y\ \overline{A^c} \subset \overline{B^c}$.

   d) $b_A = b_B \Longleftrightarrow \overline{B} = \overline{A}\ y\ \overline{A^c} = \overline{B^c} \Longleftrightarrow \overline{B} = \overline{A}\ y\ \partial A = \partial B$.

   Particularmente si $\partial\overline{A} \neq \varnothing$ se tiene

   a) $b_{\mathrm{int}A} \leqslant b_A \leqslant b_{\overline{A}}$.

   b) $b_{\overline{A}} = b_A \Longleftrightarrow \partial\overline{A} = \partial A$.

   c) $b_{\mathrm{int}A} = b_A \Longleftrightarrow \partial\mathrm{int}A = \partial A$.

2. $|b_A| = d_A + d_{A^c} = \max\{d_A, d_{A^c}\} = d_{\partial A}$

3. $\partial A = \{\ \mathbf{x} \in \mathbb{R}^n :\ b_A(\mathbf{x}) = 0\ \}$.

4. *La función b_A cumple con la condición de Lipschitz en todo su dominio.*

5. *b_A es uniformemente continua en todo $\mathbb{R}^n$.*

6. *b_A es (Fréchet) diferenciable en casi todo $\mathbb{R}^n$.*

7. *∇b_A satisface la desigualdad*

$$|\nabla b_A| \leqslant 1. \tag{2.25}$$

Al considerar el cuadrado de la función distancia con signo, la propiedad *2* del teorema anterior juega un rol particularmente importante en la caracterización de su diferenciabilidad ya que $b_A^2 = |b_A|^2 = d_{\partial A}^2$.

Teorema 2.5.15 *Sea A un conjunto con frontera no vacía y b_A su respectiva función distancia con signo.*

1. *El conjunto $\Pi_{\partial A}(\mathbf{x})$ es no vacío, compacto y para cualquier $\mathbf{x} \notin \partial A$, se tiene que*

$$\Pi_{\partial A}(\mathbf{x}) \subset \partial(\partial A).$$

2. b_A^2 *es (Fréchet) diferenciable en un punto* $\mathbf{x} \in \mathbb{R}^n$ *si y sólo si* $\Pi_{\partial A} = \{\ p_{\partial A}(\mathbf{x})\ \}$ *sólo cuenta con un punto. También se cumple*

$$p_{\partial A}(\mathbf{x}) = \mathbf{x} - \frac{1}{2}\nabla b_A^2(\mathbf{x}). \tag{2.26}$$

Observación 2.5.16 *Para* $\mathbf{x} \in \partial A$, $\Pi_{\partial A}(\mathbf{x}) = \{\ \mathbf{x}\ \}$, b_A^2 *es diferenciable en* $\mathbf{x}$ *y* $\nabla b^2(\mathbf{x}) = 0$.

Como en las funciones distancia, entre otras bondades, se puede encontrar directamente el valor del gradiente de las funciones distancia con signo, en la mayoría de los casos cuando existe.

Teorema 2.5.17 *Sea* $A \subset \mathbb{R}^n$ *con frontera no vacía.*

1. *Si* $\nabla b_A(\mathbf{x})$ *existe en un punto* $\mathbf{x} \in \mathbf{R}^n$, *entonces* $\Pi_{\partial A}(\mathbf{x}) = \{\ p_{\partial A}(\mathbf{x})\ \}$,

$$b_A(\mathbf{x}) = \begin{cases} ||p_{\partial A}(\mathbf{x}) - \mathbf{x}||, & si\ \mathbf{x} \in \operatorname{int} A^c, \\ 0, & si\ \mathbf{x} \in \partial A, \\ -||p_{\partial A}(\mathbf{x}) - \mathbf{x}||, & si\ \mathbf{x} \in \operatorname{int} A^c, \end{cases} \tag{2.27}$$

y

$$\nabla b_A(\mathbf{x}) = \begin{cases} 0 & para\ casi\ todo\ \mathbf{x} \in \partial A, \\ \frac{\mathbf{x} - p_{\partial A}(\mathbf{x})}{b_A(\mathbf{x})} & si\ \mathbf{x} \notin \partial A. \end{cases} \tag{2.28}$$

2. b_A *es diferenciable en* $\mathbf{x} \notin \partial A$ *si y solamente si* b_A^2 *es diferenciable en* $\mathbf{x}$.

3. *El esqueleto de* A *tiene medida cero en* $\mathbb{R}^n$ *y está en el exterior de la frontera de* A,

$$\operatorname{Sk}(A) = \{\ \mathbf{x} \in \mathbb{R}^n :\ \nexists \nabla b_A^2\ \} \subset \mathbb{R}^n \backslash \partial A.$$

4. *El conjunto de singularidades* ∇b_A *tiene medida cero,*

$$\operatorname{Sing}(A) = \operatorname{Sk}(A) \cup \operatorname{C}_b(A).$$

5. *Para casi todo* $\mathbf{x} \in \mathbb{R}^n$ *la función indicadora de* ∂A *se puede expresar en términos del gradiente de* b_A,

$$\chi_{\partial A}(\mathbf{x}) = 1 - ||\nabla b_A(\mathbf{x})||.$$

6. *Dado* $\mathbf{x} \in \mathbb{R}^n$, $\alpha \in [0,1]$, $\mathbf{p} \in \Pi_{\partial A}(\mathbf{x})$ *y* $\mathbf{x}_\alpha = \mathbf{p} + \alpha(\mathbf{x} - \mathbf{p})$,

$$b_A(\mathbf{x}_\alpha) = ||\mathbf{x}_\alpha - \mathbf{p}|| = \alpha||\mathbf{x} - \mathbf{p}|| = \alpha b_A(\mathbf{x}),$$

además, para cualquier $\alpha \in [0,1]$, $\Pi_{\partial A}(\mathbf{x}_\alpha) \subset \Pi_{\partial A}(\mathbf{x})$. *En particular, cuando* $\Pi_{\partial A}(\mathbf{x})$ *tiene un único elemento,* $\Pi_{\partial A}(\mathbf{x}_\alpha) = \{\ p_{\partial A}(\mathbf{x})\ \}$ *y* $\nabla b_A^2(\mathbf{x}_\alpha)$ *existe para toda* $\alpha \in [0,1]$. *Si de forma adicional* $\mathbf{x} \notin \partial A$, $\nabla b_A(\mathbf{x}_\alpha)$ *existe para* $\alpha \in (0,1]$ *y* $\nabla b_A(\mathbf{x}_\alpha) = \nabla b_A(\mathbf{x})$.

La demostración de estas propiedades se encuentra en [48]. A pesar que en la mayoría de los casos el gradiente de b_A en $\mathbf{x} \in \partial A \subset \mathbb{R}^n$ sea cero, no siempre es el caso. El conjunto de puntos donde eso ocurre es un conjunto de medida cero en $\mathbb{R}^n$, como lo indica el enunciado *1* del teorema anterior.

Teorema 2.5.18 *Sea $A \subset \mathbb{R}^n$ tal que $\partial A \neq \varnothing$. Sea $\mathbf{x} \in \partial A$, suponga que ∇b_A existe y es continua en una vecindad abierta $V(\mathbf{x})$, y que $\partial A \cap V(\mathbf{x})$ es un conjunto de medida cero. Entonces $||\nabla b_A|| = 1$ en todo el abierto $V(\mathbf{x})$.*

Cuando A es abierto y existe $\nabla b_A(\mathbf{x})$ para $\mathbf{x} \in \partial A$ y es continua en vecindades de estos punto, entonces éste siempre es un vector unitario. Este resultado no contradice al **Teorema 2.5.17**, ya que el conjunto de la frontera donde el gradiente es unitario tiene medida cero. La suavidad de ∂A se traslada a la función b_A, siendo b_A diferenciable en casi todas partes.

Observación 2.5.19 *Por todos los resultados vistos en esta sección se deduce que la función distancia con signo satisface la ecuación Eikonal en casi cualquier parte, es decir,*

$$||\nabla b_A|| = 1, \tag{2.29}$$

excepto en un conjunto de medida cero.

La función implícita ϕ que representa a la interface $\partial\Omega$ por medio del conjunto de nivel cero de una función distancia con signo es

$$\phi(\mathbf{x}) = b_{\Omega^-}(\mathbf{x}) = \begin{cases} d_{\partial\Omega^-}(\mathbf{x}), & si\ \mathbf{x} \in \text{int}(\Omega^-)^c = \partial\Omega^+, \\ 0, & si\ \mathbf{x} \in \partial\Omega^- = \partial\Omega, \\ -d_{\partial\Omega^-}(\mathbf{x}), & si\ \mathbf{x} \in \text{int}\Omega^- = \Omega^-. \end{cases} \tag{2.30}$$

Ejemplo 2.5.20 *Para la circunferencia de radio 1 centrada en el origen, su función distancia con signo es*

$$\phi(\mathbf{x}) = |\mathbf{x}| - 1, \tag{2.31}$$

y su gradiente (ver Figura 2.5)

$$\nabla\phi(\mathbf{x}) = \frac{\mathbf{x}}{|\mathbf{x}|}. \tag{2.32}$$

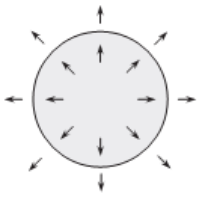


Figura 2.5: Gradiente $\nabla\phi$, para el círculo unitario centrado en el origen.

Una de las cosas más interesantes que ocurre con las funciones distancia con signo es que las funciones resultantes de las operaciones de la **proposición 2.4.4** son también funciones distancias con signo de sus respectivos conjuntos de nivel cero.

Proposición 2.5.21 *Si ϕ_1 y ϕ_2 son funciones distancia con signo, entonces también los son* mín$\{\phi_1, \phi_2\}$, máx$\{\phi_1, \phi_2\}$, máx$\{\phi_1, -\phi_2\}$ *y* $-\phi_1$.

2.6. Discretización de Interfaces y estructuras

Debido a que la representación de puntos en la computadora es finita, una figura (bidimensional o tridimensional) no puede ser representada completamente, lo único que queda es aproximarnos a ella con un número finito de puntos como muestra, lo que se conoce como discretización. En esta sección se abordará la discretización de las representaciones explícitas e implícitas de curvas y superficies, así como el de las derivadas parciales de interfaces representadas implícitamente.

Para el caso bidimensional de una curva cerrada representada paramétricamente con valores en un intervalo $[a_1, b_1]$, se escoge un conjunto de puntos $P = \{t_0, t_1, \ldots, t_s\}$ tal que $t_0 = a_1 < t_1 < t_2 < \cdots < t_s = b_1$. Los puntos de P dividen al intervalo anterior en s subintervaloes disjuntos $[t_{i-1}, t_i]$, que no necesariamente tienen el mismo tamaño $\Delta t = t_i - t_{i-1}$. El valor de la parametrización en cada punto t_i de P, denotado como $x(t_i)$, es guardado. Esto hace sencillo la discretización de la representación explícita, sin embargo la complejidad aumenta al discretizar la representación paramétrica de una superficie cerrada en el espacio tridimensional.

La estructura de orden en $\mathbb{R}$ es utilizada para establecer automáticamente las conexiones entre las imágenes de los puntos de P, de tal forma que $\mathbf{x}(t_i)$ está conectado con $\mathbf{x}(t_{i-1})$ y con $\mathbf{x}(t_{i+1})$, y los extremos del intervalo coinciden. Mientras tanto, el espacio de parámetros para discretizar una superficie es un rectángulo $[a_1, b_1] \times [a_2, b_2]$ en $\mathbb{R}^2$, el cual está desprovisto de una relación de orden. Por lo que si escogemos de forma similar a como se hizo anteriormente conjuntos $P_1 = \{r_0, \ldots, r_m\}$ y $P_2 = \{t_0, \ldots, t_n\}$ de los intervalos que conforman el rectángulo, entonces el espacio de parámetros se podría discretizar dividiendo en subrectángulos $[r_i, r_{i+1}] \times [t_j, t_{j+1}]$, con $i = 0, \ldots, m$ y $j = 0, \ldots, n$. Sin embargo, a pesar de que cada intervalo tiene un orden, por si solo el conjunto $P = P_1 \times P_2$ no da ningún indicio de los puntos $\mathbf{x}(r_i, r_j)$ que están conectados, por lo que hay que dar las conexiones explícitamente y almacenarlas. Esto lo hace aún más complejo para el caso de aplicaciones en que las conexiones cambian constantemente y de forma dinámica, como la fusión o separación de superficies, ya que es necesario hacer implementaciones adicionales para establecer nuevos enlaces, y pueden surgir conexiones que no deberían de existir, ambigüedades o huecos que no correspondan con la forma esperada de la interface dentro de la aplicación, y a la larga puede ser tedioso el trabajo para el usuario.

Para el caso bidimensional de una interface que se representa de forma implícita es necesario seleccionar una región acotada D que la contenga, denominada *dominio*, que para efectos prácticos generalmente es

un rectángulo. Este conjunto se discretiza seleccionando un conjunto finito de puntos (x_i, y_j) de él, cuyos valores de la función implícita que representa a la interface son guardados. Este conjunto se le conoce como malla, hay diferentes tipos de mallas, y una de las más utilizadas son las *mallas cartesianas*, éstas pueden ser generalizadas para discretizar dominios en espacios de más dimensiones (dominios de la forma $D = [a_1, b_1] \times \cdots \times [a_n, b_n]$ para un espacio $n-$dimensional), en lo siguiente se e definirá la versión bidimensional, pero es fácil su extensión multidimensional.

Definición 2.6.1 *Una malla cartesiana bidimensional es un conjunto finito*

$$\{(x_i, y_j) \in \mathbb{R}^2 : 1 <= i <= m, 1 <= j <= n\}$$

tal que $x_0 < \cdots < x_m$ *y* $y_0 < \cdots < y_n$. *En donde intervalos* $[x_i, x_{i+1}]$ *y* $[y_j, y_{j+1}]$ *tienen respectivamente tamaño* $\Delta x = x_{i+1} - x_i$ *y* $\Delta y = y_{i+1} - y_i$. *A cada rectángulo* $[x_i, x_{i+1}] \times [y_i, y_{i+1}]$ *se le conoce como célula y se le llama contenido a la multiplicación de sus dimensiones* $(x_{i+1} - x_i)(y_{i+1} - y_i)$. *Se dice que una malla cartesiana es uniforme si todos los tamaños de los subintervalos que se encuentran sobre un mismo eje son iguales.*

Se puede suponer $\Delta x = \Delta y$ y $(= \Delta z$, para el caso tridimensional) para estandarizar los errores de aproximación sobre cada eje. Entonces, de esta forma se determinan puntos que se encuentran en el interior, en el exterior o en la interface.

Es válido (sino obligatorio) pensar que al haber una infinidad de puntos en la interface no es posible saber de manera exacta la posición de cada uno de ellos, y sin importar qué tipo de representación se utilice siempre habrá detalles que no se puedan visualizar, pero se puede hacer más precisa la forma de la interface (es decir, la aproximación) si se aumenta el número de puntos (la resolución) que se incluyen en la discretización (en la partición del espacio de parámetros o en la malla cartesiana, según sea el caso). Para el enfoque por representación implícita, claro está, que el tamaño de un objeto debe ser más grande que el tamaño de las células de la malla, para que se pueda definir.

Una observación muy importante es que, si con la representación explícita sólo se maneja un conjunto finito de puntos de la interface, con la representación implícita, es altamente probable que la mayoría de los puntos en la malla cartesiana no estén en la interface, es decir, ella generalmente queda determinada con este tipo de representación por los puntos de la malla que se encuentran en su interior o su exterior, y algunas veces puntos de la misma interface. Aunque pueda parecer una desventaja, ya no es necesario conocer las conexiones entre puntos de la interface, por lo que la operación dinámica (como fusión o separación de objetos virtuales, tal vez simulando gotas de agua) de dos o más interfaces se hace automáticamente. A diferencia de la representación paramétrica, resulta ventajoso para la representación implícita generalizar la discretización de la interface a más de dos dimensiones, en particular para tres, de forma análoga.

Hay que ser cuidadosos con la elección de la función implícita ϕ a elegir. Mientras más suave sea esta función y menos variación haya en vecindades pequeñas cerca de la interface, entonces, estructuras

como el gradiente en puntos cercanos entre sí serán muy parecidas (con muy pocos, o nada de, cambios bruscos), por lo que se podría aproximar el gradiente de cada punto de la interface, con una interpolación de los gradientes de sus puntos vecinos en la malla cartesiana; si incluso, las normas de los gradientes no son demasiado grandes o demasiado pequeños (cercanos a cero), puede aproximarse la normal de cualquier punto de la interface con una interpolación similar. Si ciertas características, no deseadas para el cálculo de determinadas estructuras, están dadas aisladamente en unos cuantos puntos, como una singularidad (para el caso que se requiera un gradiente) o un gradiente cero (para el cálculo de una normal), se pueden definir dichas estructuras de forma similar a aquellas que hay en los puntos vecinos, ya sea escogiéndolas aleatoriamente, por conveniencia o encontrándolas por interpolación. Numéricamente, esto podría traducirse como una pequeña perturbación (que se puede dar como causa de errores de redondeo).

Ejemplo 2.6.2 *Se puede dar el caso de una circunferencia unitaria centrada en el origen cuya representación implícita puede verse en el inciso 2 del* ***Ejemplo 2.4.3****. En el punto* $(0,0)$*, se tiene* $||\nabla\phi(0,0)|| = 0$*, pero en todos los demás,* $||\nabla\phi|| = 1$*; si se quiere definir una normal en ese punto, se puede hacer según más conveniese. Podría ser* $\nabla\phi(0,0) = \nabla\phi(0,\epsilon)$ *con* ϵ *muy pequeña y no necesariamente positiva.*

Por tanto, si una función es lo suficientemente suave y bien comportada, además de que el tamaño de paso espacial sea suficientemente pequeño, el cálculo de estructuras, como derivadas, y las soluciones de ecuaciones diferenciales, como las que se utilizarán para la evolución de la interface, serán más precisas, y todo contribuirá en la estabilidad de la discretización de las ecuaciones diferenciales de movimiento. Esto hace a las *funciones distancia con signo* buenas candidatas para la representación implícita; contamos con que, en los puntos en donde no exista el gradiente o la normal, por pertenecer a un conjunto de medida cero en este sentido, puedan ser aproximada por estructuras de los puntos vecinos en donde sí existen. A veces, este tipo de funciones deben ser *reinicializadas* (como se verá en el siguiente capítulo) para que no se pierda la estructura ni las propiedades de función distancia con signo durante la evolución de la interface.

Se puede argumentar que en la discretización de la representación explícita haya un ahorro de memoria y tiempo de cálculo computacional al ser menos los puntos que se consideran (solamente puntos sobre la interface), mientras que aquella con la representación implícita, también se toman en cuenta puntos que no están en la interface. Por ejemplo, para el caso de una curva cerrada, el problema de parametrización de ésta, pasa de considerar un problema cuyo dominio es de dimensión uno, a uno en donde hay que considerar el dominio de una función implícita de dimensión más grande, de dimensión dos[17]. Sin embargo, si lo único importante es la interface (y su evolución, por supuesto), entonces, en donde se podría requerir precisión es en los puntos cercanos a ella. Por tanto se puede utilizar un método *local*, denominado *método*

[17]Un caso muy obvio para el que la aplicación de la representación implícita no es nada práctica, es en la discretización de una interface de dimensión cero (dos puntos); sólo es necesario nombrar explícitamente los puntos para tener una representación explicita, mientras que implícitamente se necesitaría discretizar un dominio $D = [a,b]$ que contuviera a los puntos de la interface o tan siquiera puntos cercanos, lo cual sería un trabajo vano.

de banda estrecha[40], para restringir los cálculos computacionales necesarios, en puntos muy cercanos a la interface y dejando a los demás puntos de la malla sobre el dominio D sin consideración. Esta técnica consiste en tomar cierta cantidad (llamada *ancho de banda*) de puntos de la malla cartesiana que se encuentren alrededor de la interface en cada dirección de los ejes, tanto en el interior como en el exterior de ésta (aunque podría ser en un sólo sentido), en donde se restringe el cálculo de estructuras (como el gradiente de la función implícita), el proceso de evolución e incluso la reinicialización a función distancia.

Al calcular estimaciones del gradiente y la curvatura media, se necesita aproximar de forma discreta a las parciales de la función ϕ. Para ello existen técnicas que utilizan los puntos de la malla cartesiana. Entre las más populares se encuentra el *método de diferencias finitas*.

Definición 2.6.3 *Sea $\phi : \mathbb{R}^2 \longrightarrow \mathbb{R}$ una función escalar y (x_k, y_l) elemento de la malla cartesiana $\{\ (x_i, y_j) : 1 \leqslant i \leqslant r, 1 \leqslant j \leqslant s\ \}$. Las aproximaciones de su derivada parcial sobre la dirección x, $\frac{\partial \phi}{\partial x}$, por diferencias hacia adelante, diferencias hacia atrás y diferencias centrales son, respectivamente,*

$$\frac{\partial \phi}{\partial x}(x_k, y_l) \approx D_x^+ \phi = \frac{\phi(x_{k+1}, y_l) - \phi(x_k, y_l)}{\Delta x} \tag{2.33}$$

$$\frac{\partial \phi}{\partial x}(x_k, y_l) \approx D_x^- \phi = \frac{\phi(x_k, y_l) - \phi(x_{k-1}, y_l)}{\Delta x} \tag{2.34}$$

$$\frac{\partial \phi}{\partial x}(x_k, y_l) \approx D_x^0 \phi = \frac{\phi(x_{k+1}, y_l) - \phi(x_{k-1}, y_l)}{2\Delta x} \tag{2.35}$$

Análogamente se definen las aproximaciones para derivadas parciales sobre la dirección y. Las fórmulas (2.31) y (2.32) tienen precisión de primer orden[18], mientras que la última fórmula tiene precisión de segundo orden. Esta definición es extensible a derivadas parciales en cada coordenada del espacio $\mathbb{R}^n$, utilizando los puntos de mallas cartesianas de dimensión n.

Para estimar la curvatura en un punto, además de necesitar las aproximaciones de las parciales, también es necesario discretizar las parciales dobles y mixtas. Una forma de hacerlo es utilizando una combinación de dos tipos de diferencias finitas, tal como:

$$\frac{\partial^2 \phi}{\partial x^2}(x_k, y_l) \approx D_x^+ D_x^- \phi = \frac{\phi(x_{k+1}, y_l) - 2\phi(x_k, y_l) + \phi(x_{k-1}, y_l)}{(\Delta x)^2}\ y \tag{2.36}$$

$$\frac{\partial^2 \phi}{\partial x \partial y}(x_k, y_l) \approx D_x^+ D_y^- \phi = \frac{\phi(x_{k+1}, y_l) - \phi(x_{k+1}, y_{l-1}) - \phi(x_k, y_l) + \phi(x_k, y_{l-1})}{\Delta y \Delta x}, \tag{2.37}$$

para puntos (x_k, y_l) de una malla cartesiana.

La función distancia con signo es suave casi donde sea, sólo está limitada por la forma de la interface

[18] El orden es el exponente menor del paso Δx de los términos no considerados de la serie de Taylor de ϕ, sobre la variable correspondiente, al despejar el término $\frac{\partial \phi}{\partial x}(x_k, x_l)$. Por ejemplo, si

$$\phi(x_k + \Delta x, x_l) = \phi(x_{k+1}, x_l) = \phi(x_k, x_l) + \frac{\partial \phi}{\partial x}(x_k, x_l)\Delta x + O((\Delta x)^2),$$

donde $O((\Delta x)^2)$ son los términos no considerados, donde Δx tiene exponente mayor o igual a dos, luego

$$\frac{\partial \phi}{\partial x}(x_k, y_l) = \frac{\phi(x_{k+1}, y_l) - \phi(x_k, y_l)}{\Delta x} + O(\Delta x).$$

Como el menor exponente de Δx en los términos de $O(\Delta x)$ es uno o mayor, por tanto, la aproximación (2.29) es de orden uno.

$\partial\Omega$. Los gradientes de la función son casi siempre de norma 1. Por tanto, el calculo de los gradientes y las normales en puntos de la interface son más fáciles de calcular que con otras funciones implícitas. Estas funciones no tienen tantas oscilaciones, es más, son crecientes mientras los puntos alejan de la interface en dirección normal hacia el exterior o decreciente si se alejan en dirección hacia el interior, por tanto son una buena opción para la representación implícita de curvas y superficies. En el siguiente capítulo, se describirá un algoritmo que servirá para encontrar funciones distancia con signo para una interface $\partial\Omega$.

Capítulo 3

Renderización háptica

Recrear artificialmente la sensación de tocar un objeto, ya sea que esta acción de tocar fuera real, pero causada de manera indirecta por un dispositivo sensible a colisiones (como cuando se maneja una herramienta a distancia para mover o cortar objetos) o ya sea que los objetos fueran simulados por computadora, se le conoce como renderización háptica. Una definición formal de este problema se puede encontrar en el libro de M. A. Otaduy [6]. En este capítulo se describe las partes básicas en las cuales se divide este problema.

La parte física más importante, es el llamado *dispositivo háptico*, el cual puede ser de diferentes diseños y formas, pero uno de sus principales características son los grados de libertad[1] (DOF, por las siglas de Degree of Freedom), generalmente contando con 3 DOF y 6 DOF, lo que hay que tomar en cuenta en la parte virtual. Para una mejor usabilidad y rendimiento del sistema se sugiere que la configuración de la herramienta virtual se parezca mucho a la del dispositivo háptico, por lo cual deben tener el mismo número de grados de libertad, y en la medida de lo posible mientras más simple, mejor. Por ejemplo, pensando en un ambiente tridimensional cuando una herramienta que consiste de un punto simple sólo puede contar a lo más de 3 DOF, teniendo la libertad de moverse en tres direcciones; en el caso de un objeto tridimensional más complejo (rígido y sin articulaciones) puede tener 6 DOF, tres parámetros para la posición y tres para su orientación. También se debe hacer un acople del dispositivo háptico con la herramienta virtual, tal que el movimiento de uno refleje el movimiento del otro, y una manera habitual de hacerlo es acoplando el centro de masa de la herramienta con el movimiento del dispositivo.

Un algoritmo de renderización háptica puede ser visto como una *impedancia mecánica programable*, el sistema de control del dispositivo háptico proporciona posiciones que son utilizadas para implementar un ciclo de control de fuerza, esto es en el caso en que los datos de entrada del sistema háptico midan posición para generar datos de salida de fuerza, que deben ser controlados para la estabilidad del sistema físico. Inversamente, si los datos de entrada son fuerzas y los de salida posiciones, un algoritmo de

[1] Los grados de libertad son el número mínimo de velocidades generalizadas independientes necesarias para definir el estado cinemático de un mecanismo o sistema mecánico.

renderización háptica puede implementarse como una *admitancia programable*, en donde el control del dispositivo proporciona fuerzas para implementar un ciclo de control de posición. Como se tiene pensado que a partir de las posiciones y movimientos introducidos por el usuario se obtenga una fuerza resultante, en la aplicación de un sistema háptico completo, el enfoque de impedancia es el que conviene. Aquí se acaba de mencionar uno de los principales componentes de los algoritmos de renderización háptica, que son los *algoritmos de control*, sin embargo su implementación no es uno de los objetivos del trabajo por lo que sólo se hará mención de ellos como parte de la renderización háptica.

Los tres principales componentes que tienen los algoritmos de renderización háptica son: el módulo de *detección de colisiones*, donde un algoritmo se encarga de proporcionar información sobre el contacto entre un objeto del ambiente y una herramienta virtual, el módulo de *respuesta de colisión*, aquí se calcula la fuerza que incide en la herramienta virtual, y finalmente el módulo de control, que envía una fuerza hacia el dispositivo háptico, para retroalimentar al usuario, tratando de aproximar la fuerza calculada en los algoritmos anteriores, y adaptándola a la mejor opción dentro de las capacidades del dispositivo háptico y paraevitar inestabilidades como vibraciones o cambios muy bruscos de fuerzas. En este trabajo, la idea de introducir la renderización háptica es con el objetivo de encontrar alguna manera de poder adaptar la representación de función distancia con signo para llevar a cabo la detección de colisión de un objeto representado de forma explícita en un ambiente representado de forma implícita,y además modelar las fuerzas de contacto al haber una colisión entre dicho objeto explícito con los objetos del ambiente. Se presentarán los algoritmos necesarios que serán utilizados para llevar a cabo la renderización háptica.

3.1. Detección de colisiones

Este módulo además de ser parte de la renderización háptica, también es importante en los sistemas de escultura virtual, pues detecta el momento en que la herramienta ha tenido contacto con el objeto a modelar, y luego se lleva a cabo la deformación local y el cálculo de la fuerza de contacto. La colisión entre una herramienta virtual y los objetos del ambiente resulta más fácil cuando alguno de los dos tiene una representación implícita. Es más sencillo identificar una colisión definiendo el interior y el exterior de al menos uno de ellos. Debido a que es difícil encontrar el momento exacto del primer contacto entre dos cuerpos, es más probable que se determine cuando ya haya ocurrido una penetración entre ambos. ¿De qué manera?

Se supone que se tiene una herramienta, la cual es definida explícitamente, discretizada con un cierto número de puntos, y se mueve dentro del ambiente virtual, que es el dominio de una función implícita ϕ, que está discretizada con una malla cartesiana uniforme. Esta función representa implícitamente a una interface con su conjunto de nivel cero. Para los puntos de la herramienta $\hat{\mathbf{x}}_k$ que conforman la discretización de la herramienta, se pueden aproximar los valores de la función implícita correspondientes a cada uno de ellos, a través de una interpolación, utilizando para ello los valores de los puntos de la celda

de malla en que se encuentran. Para el caso bidimensional, mostrado en la Figura 3.1, una estimación por interpolación *bilineal* (para el caso tridimensional sería una interpolación *trilineal*) se encontraría, buscando el valor de la función distancia de dos puntos intermedios en cada celda, $\mathbf{q}_1$ y $\mathbf{q}_2$, situados en las caras horizontales pero con la misma abscisa que el punto $\hat{\mathbf{x}}_k$ en cuestión (aunque nada impide también hallar los puntos correspondientes en las caras verticales con la misma ordenada que $\hat{\mathbf{x}}_k$).

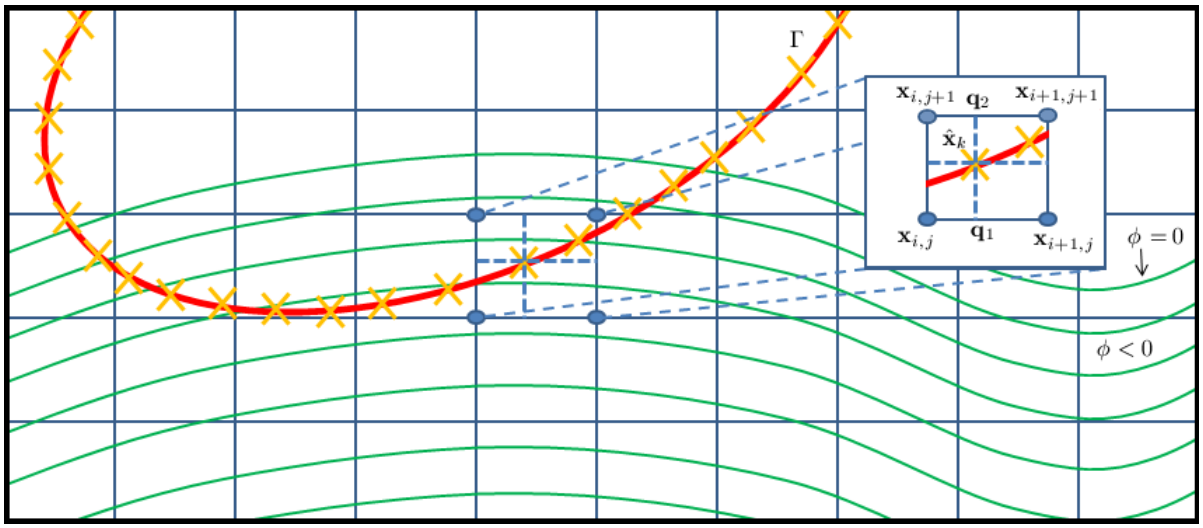


Figura 3.1: Detección de colisión. La región que ocupa la herramienta es denotada por Γ, su frontera es la curva de color rojo y los puntos discretizados de la herramientas son las cruces color amarillo. Por otro lado, las curvas verdes son los conjuntos de nivel correspondientes a valores no nulos de la función ϕ que representa el material deformable.

Si $\mathbf{q}_1$ tiene como vecinos de malla a $\mathbf{x}_{i,j} = (x_i, y_j)$ y a $\mathbf{x}_{i+1,j} = (x_{i+1}, y_j)$, y $\mathbf{q}_2$ tiene como vecinos a $\mathbf{x}_{i,j+1} = (x_i, y_{j+1})$ y a $\mathbf{x}_{i+1,j+1} = (x_{i+1}, y_{j+1})$, entonces sus valores estarían calculados por

$$\phi(\mathbf{q}_1) \approx \frac{x_{i+1} - \hat{x}_k}{x_{i+1} - x_i}\phi(\mathbf{x}_{i,j}) + \frac{\hat{x}_k - x_i}{x_{i+1} - x_i}\phi(\mathbf{x}_{i+1,j})$$

y

$$\phi(\mathbf{q}_2) \approx \frac{x_{i+1} - \hat{x}_k}{x_{i+1} - x_i}\phi(\mathbf{x}_{i,j+1}) + \frac{\hat{x}_k - x_i}{x_{i+1} - x_i}\phi(\mathbf{x}_{i+1,j+1}),$$

tal que $\hat{\mathbf{x}}_k = (\hat{x}_k, \hat{y}_k)$. De aquí se encuentra el valor de $\phi(\hat{\mathbf{x}}_k)$,

$$\phi(\hat{\mathbf{x}}_k) \approx \frac{y_{j+1} - \hat{y}_k}{y_{j+1} - y_j}\phi(\mathbf{q}_1) + \frac{\hat{y}_k - y_j}{y_{j+1} - y_j}\phi(\mathbf{q}_2),$$

esto es,

$$\phi(\hat{\mathbf{x}}_k) \approx \frac{1}{(x_{i+1} - x_i)(y_{j+1} - y_j)}(\phi(\mathbf{x}_{i,j})(x_{i+1} - \hat{x}_k)(y_{j+1} - \hat{y}_k) + \phi(\mathbf{x}_{i+1,j})(\hat{x}_k - x_i)(y_{j+1} - \hat{y}_k) + ...$$
$$... + \phi(\mathbf{x}_{i,j+1})(x_{i+1} - \hat{x}_k)(\hat{y}_k - y_j) + \phi(\mathbf{x}_{i+1,j+1})(\hat{x}_k - x_i)(\hat{y}_k - y_j)). \quad (3.1)$$

Luego que se hayan encontrado dichos valores, pueden ser analizados; si al menos, uno tiene un valor negativo o nulo, entonces hubo una colisión, si de lo contrario, todos fueron positivos, no ha ocurrido ningún contacto y la herramienta puede seguir su curso, haciendo el procedimiento anterior durante todo su trayecto hasta que sea detectada una colisión.

Algunos problemas pueden surgir, como cuando la herramienta traspasa los objetos del ambiente sin siquiera haberse detectado las correspondientes colisiones, ya sea que pase demasiado rápido a través de ellos, que no haya detectado partes delgadas de éstos, que la interface haya quedado contenida dentro de la herramienta (esto puede ocurrir si la herramienta es más grande que el objeto y que sólo su frontera es discreizada para análisis de detección) o que pase por alguna punta. Para tratar de solventar este problema existen los llamados *algoritmos de detección continua*, en la referencia [7] puede encontrarse una buena explicación de este tipo de algoritmos. En el caso en que la herramienta que se considera sólo tenga la posibilidad de trasladarse de un lugar a otro, sin rotaciones, puede considerarse algo relativamente sencillo (aunque aumente algo el costo computacional) para no pasar por alto algunas colisiones: determinar segmentos de rectas entre la posición inicial y la posición final de los movimiento de traslación de cada punto de la herramienta. Al discretizar estas líneas, se aproximan los valores de la función implícita de cada uno de sus puntos de la misma manera que se hace con los de la herramienta, y se proceden a analizar para saber si en las posiciones intermedias pudo haberse dado alguna colisión. Al confirmarse la colisión, los puntos de la herramienta podrían posicionarse de tal manera que retrocedan en la posición a la que debieron haber estado en el momento del primer contacto.

3.2. Respuesta de colisión

En este módulo, se calculan las fuerzas que inciden en la herramienta virtual. Los dos tipos de percepción háptica que se tratan de emular en los algoritmos de renderización háptica es la percepción táctil y la kinética, ya sea a parte o combinadas. Las fuerzas del primer tipo pueden ser calculadas por medio de adaptaciones de leyes físicas como la *ley de coulomb*[8]. Pero aquí solamente se toman en cuenta las fuerzas generadas al hacer presión sobre un objeto. La forma de hacerlo es calcular las fuerzas normales a la superficie del objeto deformable que ejercen sobre la herramienta con el contacto. Para esto, cuando se detecta una colisión y se determinan los puntos (discretos) $\hat{\mathbf{x}}_k$ de la herramienta que están en el interior de la interface, en cada uno de ellos se encuentra la fuerza $\mathbf{F}_k$ que ejerce sobre cada uno.

Para encontrar la dirección de cada $\mathbf{F}_k$, se encuentra el gradiente de la función implícita por medio de la interpolación de los gradientes que conforman la celda en donde se encuentra el punto $\hat{\mathbf{x}}_k$ (ver Figura 3.2), lo cual se justifica debido al inciso *5* de la **Proposición 2.5.10**, los gradientes de la función distancia con signo en estos puntos vecinos, tienen la misma dirección que la normal de puntos muy cercanos a la interface, cuando dichas estructuras existen; además los gradientes son unitarios y existen en casi todas partes. Se puede utilizar un procedimiento similar al de hallar el valor de función distancia de un punto

de la herramienta interpolando linealmente los valores de los puntos de la celda en donde se encuentra contenido. En el caso bidimensional, el gradiente de la función implícita en un punto $\hat{\mathbf{x}}_k$ de la herramienta es

$$\nabla\phi(\hat{\mathbf{x}}_k) \approx \frac{1}{(x_{i+1}-x_i)(y_{j+1}-y_j)}(\nabla\phi(\mathbf{x}_{i,j})(x_{i+1}-\hat{x}_k)(y_{j+1}-\hat{y}_k) + \nabla\phi(\mathbf{x}_{i+1,j})(\hat{x}_k-x_i)(y_{j+1}-\hat{y}_k) + ...$$

$$... + \nabla\phi(\mathbf{x}_{i,j+1})(x_{i+1}-\hat{x}_k)(\hat{y}_k-y_j) + \nabla\phi(\mathbf{x}_{i+1,j+1})(\hat{x}_k-x_i)(\hat{y}_k-y_j)). \tag{3.2}$$

Similarmente se procede para tres dimensiones.

También se debe calcular la magnitud de las fuerzas que inciden en cada punto de la herramienta dentro del objeto. Los métodos más populares para hallar la magnitud de la fuerza utilizan la *ley de Hooke*, calculando la distancia de penetración de la herramienta dentro del objeto del ambiente, medido en dirección normal a la interface, esto es, el valor de la función distancia de los puntos de la herramienta que penetraron al objeto. Entonces la fuerza en $\hat{\mathbf{x}}_k$, queda como

$$\mathbf{F}_k = \kappa\phi(\hat{\mathbf{x}}_k)\nabla\phi(\hat{\mathbf{x}}_k) \tag{3.3}$$

donde κ es la constante de rigidez o elasticidad, dependiendo de que tan elástico o rígido sea el material del objeto con el que se está teniendo colisión. A partir de estas fuerzas normales se calcula la dirección y la magnitud de la fuerza que experimentará el centro de masas de la herramienta y la cual se tratará de reproducir para la retroalimentación del usuario por parte del dispositivo físico. Si m es el número de elementos $\hat{\mathbf{x}}_k$ de la herramienta que intersectan o están en el interior de la interface, como $\mathbf{F}_k$ es la fuerza generada en cada uno de esos puntos, entonces la fuerza total puede ser definida como

$$\mathbf{F} = \sum_{k=1}^{m} \mathbf{F}_k. \tag{3.4}$$

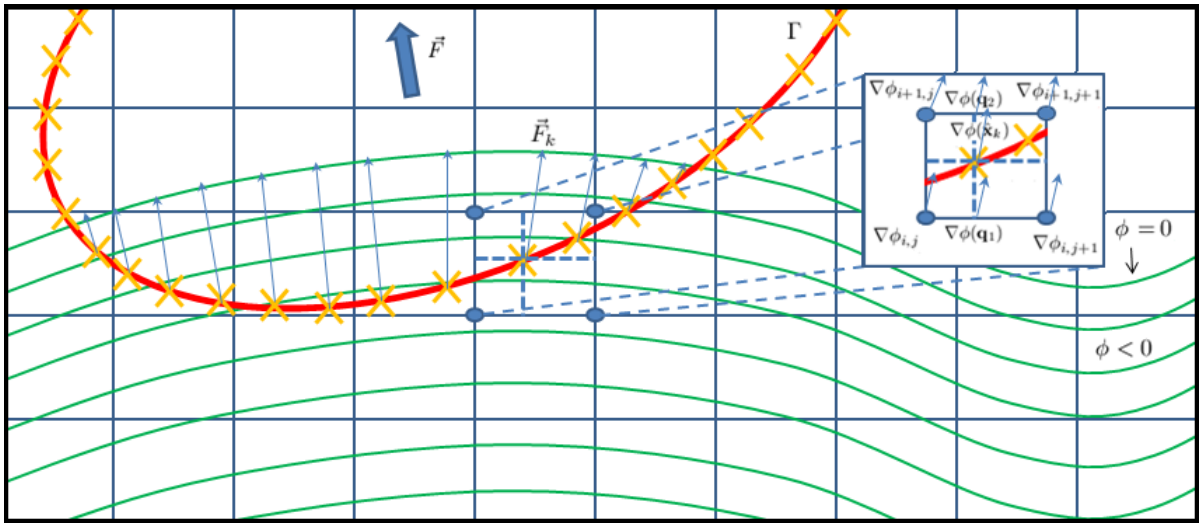


Figura 3.2: Respuesta de colisión. La figura es análoga a la anterior, excepto que en cada punto discretizado de la herramienta se ha dibujado la fuerza que incide sobre él, hay que aclarar que el número κ en este caso se ha escogido como 1. En el recuadro se puede observar los gradientes de la función ϕ de los vecinos de malla del punto $\hat{\mathbf{x}}_k$), así como el suyo mismo. La fuerza $\mathbf{F}$ es la resultante que incide sobre el centro de masas de la herramienta.

Sin embargo, como la igualdad (5.3) así tal cual puede generar fuerzas muy grandes, pudiendo generar inestabilidades, puede utilizar el promedio de las fuerzas. Si n es el número total de puntos de la herramienta dentro o en la superficie de un objeto del ambiente virtual, entonces

$$\mathbf{F} = \frac{1}{n}\sum_{k=1}^{m}\mathbf{F}_k. \tag{3.5}$$

se utiliza para hallar la fuerza. Otra posible fuente de inestabilidad es utilizar valores de κ muy altos, pero para los propósitos de la tesis, ya que se quiere que el material del objeto sea deformable, la rigidez κ no debe ser demasiado grande.

3.3. Implementación de la renderización háptica

Mientras que la herramienta no colisione con ningún objeto del ambiente es libre de moverse en cualquier dirección. En todo momento de su recorrido en el espacio, han de analizarse sus puntos, hasta detectar la colisión, es decir, el módulo de detección de colisión siempre está presente, y luego se calculan las fuerzas en respuesta a la colisión como previamente se ha mostrado, para luego ser enviado al usuario como una retroalimentación de fuerza.

Esto ha sido la descripción de los componentes de los componentes de detección de colisión y de cálculo de fuerzas de contacto de la renderización háptica, con la ayuda de una representación implícita de función distancia con signo, que son utilizados en este trabajo. En el siguiente capítulo se empezará a

Algoritmo 1 Renderización háptica

Entrada: Los valores de función ϕ que representa al material deformable, los puntos P_H de la representación explícita de la herramienta virtual. OBSERVACIÓN: Se supone que la herramienta se mueve libremente en un espacio bidimensional o tridimensional.

Salida: La fuerza resultante **F** y el número de puntos de colisión, *colision*.

1: Se inicializa el número de puntos de colisión, *colision*, a cero.
2: **mientras** que *colision*=0 **hacer**
3: **para** $i = 1 : n$, donde n es el número de puntos de la herramienta. **hacer**
4: Se encuentra el valor de ϕ del punto de la herramienta P_H correspondiente, por medio de la interpolaión (3.1).
5: **si** el valor ϕ del punto de P_H de la herramienta es no positivo **entonces**
6: Aumenta el valor de *colision* en 1.
7: **fin si**
8: **fin para**
9: **fin mientras**
10: **si** *colision*$\neq$ 0 **entonces**
11: Se encuentran los gradientes de ϕ en cada punto de la herramienta con valores de ϕ no positivos con (3. 2).
12: Se encuentra el valor de la fuerza en cada uno de los puntos de la herramienta en colisión utilizando el gradiente de cada uno de esos puntos y los valores de ϕ correspondientes encontrados en el paso 3, aplicando la ecuación (3.3).
13: Con (3.4) ó (3.5) se calcula la fuerza resultante.
14: **si no**
15: $\mathbf{F} = \mathbf{0}$
16: **fin si**
17: **devolver** la fuerza resultante **F**.

abordar los temas de la deformación de la interface, en donde también se aprovechará la representación implícita de función distancia y por último se integrará la parte de cálculo de fuerzas a la parte de deformación.

Capítulo 4

Métodos de conjuntos de nivel

Ya se determinó la forma de representación de las superficies que definirán a los objetos, particularmente los cuerpos deformables se representarán a través de una función implícita, idealmente que sea lo más suave posible para aprovechar estructuras como el gradiente. Una buena opción es la función distancia con signo, en este capítulo se vislumbrará cómo este tipo de funciones puede ayudar a simplificar o mejorar el movimiento de la interface. Recordar que se desea recrear la deformación de un objeto maleable cuando otro hace presión sobre él, moviendo parte del material hacia su interior y de algún modo evitar la variación del volumen total del cuerpo, para esto se requerirá el movimiento o evolución de su frontera (una curva o superficie). Con este objetivo se presenta una herramienta muy importante la cual aprovecha la representación implícita de la interface que define al objeto deformable.

4.1. Ecuación de conjunto de nivel

Suponga que se tiene un cuerpo (en dos o tres dimensiones) Ω, cuyo interior, exterior y su frontera son, respectivamente, Ω^-, Ω^+ y $\partial\Omega$ Como se supone que este último conjunto está representado implícitamente, entonces también se debe considerar una función ϕ, cuyo conjunto de nivel cero sea dicha superficie ($\partial\Omega = \{\mathbf{x} \in \mathbb{R}^3 : \phi(\mathbf{x}) = 0\}$).

Sea $\mathbf{x} \in \mathbb{R}^3$. Este punto con el correr del tiempo describe una recorrido, y suponga que la función de esta trayectoria tiene primera derivada con respecto al tiempo:

$$p(\mathbf{x}, \cdot) : [0, \infty) \longrightarrow \mathbb{R}^3$$

$$t \mapsto p(\mathbf{x}, t) = (x(t), y(t), z(t)), \tag{4.1}$$

tal que $p(\mathbf{x}, 0) = (x(0), y(0), z(0)) = \mathbf{x}$. Como en cada tiempo t los valores de la función implícita deben cambiar, entonces para derivar la fórmula de movimiento de los conjuntos de nivel de ϕ se necesita

encontrar una función que dependa del espacio $\mathbb{R}^3$ y del tiempo t. Esta función es la que describirá el comportamiento o movimiento de la gráfica de ϕ.

$$\Phi : \mathbb{R}^3 \times [0, \infty) \longrightarrow \mathbb{R}$$

$$(\vec{x}, t) \mapsto \Phi(p(\mathbf{x}, t), t), \tag{4.2}$$

tal que $\Phi(\mathbf{x}, 0) = \phi(\mathbf{x})$ para cualquier $\mathbf{x}$. Sin embargo, el punto $\mathbf{x}$ debe permanecer en el mismo conjunto de nivel en toda su trayectoria, esto es, el valor de Φ en la trayectoria de $\mathbf{x}$ es constante, $\Phi(p(\mathbf{x}, t), t) = \phi(\mathbf{x})$. De aquí, al derivar con respecto al tiempo y haciendo uso de la regla de la cadena se tiene la siguiente expresión

$$\frac{d\Phi}{dt}(p(\mathbf{x}, t)) = \left(\Phi_x \frac{dx}{dt} + \Phi_y \frac{dy}{dt} + \Phi_z \frac{dz}{dt}\right) + \Phi_t = \nabla\Phi \cdot \mathbf{V} + \Phi_t = 0, \tag{4.3}$$

con $\mathbf{V} = \left(\frac{dx}{dt}, \frac{dy}{dt}, \frac{dz}{dt}\right)$ y, denotando la parte espacial del gradiente de Φ como $\nabla\Phi(p(\mathbf{x}, t), t) = \nabla\phi(p(\mathbf{x}, t))$ tal que $\nabla\Phi(p(\mathbf{x}, 0), 0) = \nabla\phi(\mathbf{x})$; en adelante, sin riesgo de confusión se denotará a $\nabla\Phi(p(\mathbf{x}, t), t)$ por $\nabla\phi(\mathbf{x})$ y a Φ por ϕ. A esta ecuación diferencial se le conoce como la ecuación de conjunto de nivel, pero hay que tener en cuenta que el campo de vectores $\mathbf{V}$ debe ser conocido, en primera instancia, para todo punto en el dominio de la función implícita. Aunque en este caso se esté trabajando sobre el espacio $\mathbb{R}^3$, el argumento puede ser extendido a espacios de un número cualquiera de dimensiones.

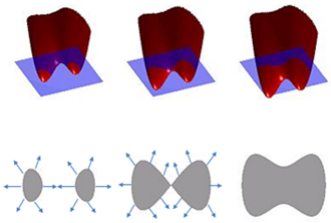


Figura 4.1: En la parte superior se muestra la gráfica de ϕ, cortada por el plano $z = 0$. Abajo se se muestra el conjunto de nivel cero y su interior. Las flechas indican la dirección de movimiento de la interface.

4.2. Campos de velocidades

El campo vectorial de velocidades $\mathbf{V}$ indica la velocidad con la que se moverán los puntos del dominio de la función implícita ϕ en un tiempo determinado. Este campo entonces puede ser visto como una función que depende de puntos del espacio (bidimensional o tridimensional) y del tiempo, con valores en el espacio ambiente $\mathbf{V} = \mathbf{V}(p(\mathbf{x}, t), t)$. A pesar de que dicha función está definida en todo el dominio la velocidad que determina la evolución de la interface es precisamente la velocidad dada a cada uno de sus puntos, la definida para puntos fuera de la interface realmente no tienen ningún significado para ella, por lo que dada $\mathbf{V}$ a lo largo del conjunto de nivel cero, una estrategia razonable para definir la velocidad en los puntos fuera de ella es que sea constante, aunque alternativamente se podría definir la velocidad en

un punto $\mathbf{x} \notin \partial\Omega$ como la velocidad que tiene el punto más cercano $\mathbf{x}_C$ de la interface , $\mathbf{V}(\mathbf{x}) = \mathbf{V}(\mathbf{x}_C)$. De hecho, para efectos numéricos sería recomendable que puntos alrededor de la interface sean definidas de esta última forma.

La velocidad puede estar determinada por una regla física o puede depender directamente de la forma geométrica de la interface (denominados *campos de velocidades autogenerados* o *intrínsecamente generados*). En este trabajo es de interés el segundo caso para modelar la deformación del objeto virtual. Las velocidades intrínsecamente generadas presentan la forma $\mathbf{V}(\mathbf{x}) = b\nabla\Phi(\mathbf{x})/||\nabla\Phi||$, donde b es un escalar y puede ir variando en todo el dominio. La ecuación (4.3) se convierte en

$$\nabla\phi \cdot \mathbf{V} + \phi_t = (b\nabla\phi \cdot \nabla\phi)/||\nabla\phi|| + \phi_t = b||\nabla\phi|| + \phi_t = 0. \tag{4.4}$$

Con este campo de velocidades los puntos de la interface se mueven en dirección normal con respecto a la misma interface, por esto se dice que dependen de la forma que tiene ésta, y un aspecto a destacar es que cuando b es constante se cumple que la velocidad de cualquier punto coincide con la velocidad del punto más cercano a la interface; aquí cabe hacer notar que cuando $b > 0$, la interface se mueve en dirección hacia su exterior, si $b = 0$, permanece estática ($\phi_t = 0$), mientras que con $b < 0$, se dirige hacia su interior, contrayéndose. Un caso particular de (4.4) y que es muy utilizado entre las aplicaciones de los métodos de conjuntos de nivel, es cuando $b = -a\kappa$, donde a es constante y κ es la curvatura media, en cuyo caso mientras más grande sea el valor absoluto de la curvatura los puntos se moverán a mayor velocidad. Si la función ϕ es todo el tiempo una función distancia con signo (o muy cercano de serlo) entonces la ecuación (4.4) se simplificaría aún más: $\phi_t = -b$, o en su caso $\phi_t = -a\kappa$; lo que motiva aún más el uso de este tipo de funciones. Para saber más sobre aplicaciones o estudios donde se utilizan campos de velocidades que dependen de la curvatura media puede verse en las referencias [40, 49].

4.3. Discretización de la ecuación de conjunto de nivel

Suponga que tanto ϕ como $\mathbf{V}$ cuyos valores se conocen en una malla cartesiana de un dominio rectangular (o al menos en una banda estrecha alrededor de la interface). Al tiempo actual (pudiendo no ser el tiempo inicial) y se escribe como t^n, y el valor de la función implícita ϕ en ese tiempo se le denota ϕ^n. Actualizar ϕ significa hallar su valor en un tiempo $t^{n+1} = t^n + \Delta t$, denotado como $\phi(t^{n+1})$.

Para cualquier versión de la ecuación(4.3) las parciales espaciales y temporales pueden tomarse por separado. Para la derivada parcial con respecto al tiempo puede utilizarse el *método de Euler hacia adelante*, primer orden de precisión con respecto al tiempo,

$$\frac{\phi_{i,j}^{n+1} - \phi_{i,j}^n}{\Delta t} = -\mathbf{V}_{i,j}^n \cdot \nabla\phi_{i-j}^n, \tag{4.5}$$

donde $\mathbf{V}_{i,j}^n$ y $\nabla\phi_{i,j}^n$, representan los valores actuales del campo de velocidades y del gradiente de ϕ en el punto $\mathbf{x}_{i,j} = (x_i, y_j)$, respectivamente. Esta es una de las formas más sencillas de discretizar la parte temporal de una ecuación diferencial parcial, pero también pueden utilizarse otros métodos, ya sea de

una manera implícita como el *método Euler hacia atrás*, o para aumentar el grado de precisión de la discretización, con los *métodos de TVD Runge-Kutta* desarrollados por Shu y Osher en [50].

Pasando a la discretización de las parciales espaciales hay que tener en cuenta la dirección hacia la cual se mueve la información. Al expandir la ecuación (4.5), queda como

$$\frac{\phi_{i,j}^{n+1} - \phi_{i,j}^{n}}{\Delta t} = -u_{i,j}^{n}(\phi_x)_{i,j}^{n} - v_{i,j}^{n}(\phi_y)_{i,j}^{n}, \tag{4.6}$$

tal que $\mathbf{V}_{i,j}^{n} = (u_{i,j}^{n}, v_{i,j}^{n})$. Atendiendo al *método de las características*, para discretizar $u_{i,j}^{n}(\phi_x)_{i,j}$, se debe analizar el signo de $u_{i,j}^{n}$. Si $u_{i,j}^{n} > 0$, entonces la información va de izquierda a derecha en $\mathbb{R}^2$, por lo que hay que fijarse de los valores de ϕ a la izquierda de $\mathbf{x}_{i,j}$, que influirán en su valor en el siguiente tiempo, por lo que la mejor opción sería utilizar $D_x^{-}\phi$ para discretizar $(\phi_x)_{i,j}^{n}$, ya que se cuenta con dicha información, lo cual no cuenta $D_x^{+}\phi$; en caso de que $u_{i,j}^{n} < 0$, entonces la información va en dirección contraria y por tanto la opción a utilizar sería $D_x^{+}\phi$; por último, cuando $u_{i,j} = 0$, el término correspondiente a $(\phi_x)_{i,j}^{n}$ desaparece, por lo que no es necesario utilizar una fórmula numérica. Análogamente, se tiene que hacer el mismo procedimiento para las otras coordenadas espaciales. Este método es denominado *contraviento* (*upwinding*, en inglés), y es una discretización de primer orden de la parte espacial, ya que para todas las coordenadas tanto $D^{-}\phi$ como $D^{+}\phi$ son aproximaciones de primer orden para la derivada parcial. Hay que aclarar que en este caso se supone que V, no la conforma o no depende de derivadas parciales de ϕ de orden mayor o igual a dos, como lo hace la velocidad compuesta por la curvatura media. Para este caso se utilizan otros métodos correspondientes para la solución de *ecuaciones parciales hiperbólicas*[10].

La *estabilidad* es una propiedad que garantiza que los errores pequeños en la aproximación no sean amplificados mientras la solución evoluciona en el tiempo. Una condición para ser satisfecha es la *condición de Courant-Friedreichs-Lewy* (la condición CFL, como normalmente se conoce). Lo que afirma es que, una onda numérica se propaga al menos tan rápido como las ondas físicas. Por ejemplo, para el caso de una dimensión la velocidad de onda numérica $\Delta x/\Delta t$ debe ser al menos tan rápido como la velocidad de onda física $|u_{i,j}^{n}|$ para todos los puntos de la red cartesiana, es decir, $\Delta x/\Delta t > max_{\forall i,n}\{|u_i^n|\}$. Esto lleva al número de CFL, $\alpha = \Delta t(max_{\forall i,n}\{|u_i^n|/\Delta x\})$ para el caso de una dimensión, $\alpha = \Delta t(max_{\forall i,j,n}\{|u_{i,j}^{n}|/\Delta x + |u_{i,j}^{n}|/\Delta y\})$ para dos dimensiones y análogamente para tres, aunque otro número de condición utilizado es $\alpha = max\{||\mathbf{V}||\}/min\{\Delta x, \Delta y, \Delta z\}$. Como afirma la condición de CFL, se debe cumplir con $\alpha \in (0, 1)$.

Ahora bien, si se desea utilizar algo más de precisión numérica en la discretización parcial, entonces en lugar de utilizar el método de contraviento, se puede tomar la fórmula de diferencias centrales con el que da una discretización de segundo orden de precisión. Sin embargo, con el método de Euler hacia adelante no es estable con las condiciones usuales de CFL con $\Delta t \sim \Delta x$, pero según [49] se puede alcanzar con una condición de CFL más restrictiva, tal que $\Delta t \sim (\Delta x)^2$.

4.4. Reinicialización

Durante la evolución de la interface, con el transcurrir del tiempo, la función distancia con signo inicial que la representa generalmente va peridendo su estructura; ya sea por la misma definición del vector velocidad $\mathbf{V}$ o por la acumulación de errores de redondeo de la discretización del movimiento de la interface. La uniformidad de los conjuntos de nivel de este tipo de funciones, sobre todo, alrededor y a lo largo de la interface, hace más precisa la solución numérica de la ecuación de movimiento, pero la aparición de fenómenos como el apilamiento de conjuntos de nivel así como la gran separación de estos en lugares muy cercanos al nivel cero, puede causar eventualmente imprecisiones aún mayores, que vuelva inestable al movimiento de la interface.

Con el fin de conservar la estructura de conjuntos de nivel de la función distancia con signo se ha creado un concepto denominado *reinicialización*, introducido por Chopp en [51]. La reinicialización consiste en aplicar periódicamente un método de construcción de funciones distancia en el transcurso de la evolución de la interface. Hay que tener en cuenta que la precisión y la frecuencia de las técnicas de reinicialización aumenta el tiempo de cálculo de resolución de movimiento, por lo que sería muy conveniente aplicar ambos procesos (reinicialización y evolución de movimiento) a regiones muy cercanas a la inerface sobre todo si lo que único que interesa es el conjunto de nivel cero; los problemas que se puedan tener lejos de este conjunto no afectaría en los resultados.

Uno de los métodos más utilizados de reinicialización aprovecha una de las propiedades más importantes de la función distancia con signo, la de tener un gradiente unitario casi donde sea, $||\nabla\phi|| = 1$. La idea consiste en que si una función deja de cambiar o evolucionar en el tiempo implica que $\phi_t = 0$ (derivada parcial en tiempo), entonces resolviendo la ecuación diferencial

$$\phi_t + ||\nabla\phi|| = 1, \tag{4.7}$$

se podrá construir una función distancia con signo con esa característica que será la función distancia con signo. Cuando la parcial en el tiempo se anula se le conoce como un *estado estable*. La ecuación (4.7) mueve a la interface en dirección norma así misma, la infomación fluye de los valores más pequeños de ϕ a los mayores, por lo cual el valor de la función distancia de los últimos depende de los primeros. Esta ecuación no es utilizada frecuentemente para la reinicialización, ya que los valores negativos de la función que se reinicializa pueden influir y modificar el conjunto de nivel cero, es decir puede cambiar la interface. Una forma de resolver este problema es dividir al espacio en dos, en el interior y el exterior de la interface, como se ha visto antes. Luego se calculan los valores de la función distancia con signo a los puntos adyacentes a la interface directamente, mano a mano o con una técnica especializada para ello. Estos puntos adyacentes funcionan como condiciones de frontera, para funciones diferenciales que se aplicarán a cada una de las regiones por separado. Por ejemplo, para la parte exterior, se utiliza la ecuación (4.7), utilizando los valores de los puntos adyacentes en el exterior se procede a encontrar las imágenes de la función distancia con signo (en este caso los valores son positivos), yendo hacia fuera de

la interface, sin afectar el conjunto de nivel cero. Para el interior se procede de manera similar, pero solucionando la ecuación (4.8).

$$\phi_t - ||\nabla\phi|| = -1. \tag{4.8}$$

Sussman, Smereka y Osher, introdujeron en [57] una ecuación que introducía, tanto la ecuación (4.7) como la (4.8) e una sola. Esta ecuación es conocida como ecuación de reinicialización:

$$\phi_t + S(\phi_0)(||\nabla\phi|| - 1) = 0. \tag{4.9}$$

$S(\phi_0)$ es una función signo cuya imagen es 1 en el exterior, -1 en el interior y cero en la interface, que inicialmente se representa por una función inicial ϕ_0. Esta función distancia tiene algunas variantes, entre ellas se encuentra la función (4.10) sugerida en [57], como una forma suave numéricamente

$$S(\phi_0) = \frac{\phi_0}{\sqrt{\phi_0^2 + (\Delta x)^2}}. \tag{4.10}$$

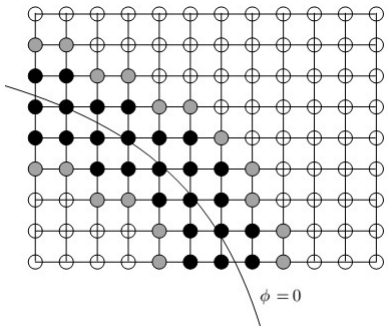


Figura 4.2: Puntos adyacentes.

En este trabajo se utiliza la función de matlab reinit_SD, que calcula la función distancia con signo de una función utilizando precisamente la ecuación de reinicialización con función signo (4.10). Esto servirá para llevar a cabo algunos experimentos en la parte de deformación y saber qué tan conveniente puede ser su implementación en los algoritmos.

Ahora, en el siguiente capítulo se expondrá un modelo de deformación, que derivará en una expresión de conjuntos de nivel que hará mover a una interface, conservando el volumen de su interior.

Capítulo 5

Modelo de deformación local con volumen constante

Como se dijo desde el principio de la tesis, uno de los objetivos de ella es encontrar un modo de deformar localmente la interface, tal que el volumen o el área de su interior, según sea el caso, no cambie. Cuando se habla de deformación local, se supone que el cuerpo de la herramienta tiene contacto con el material deformable, moviéndose hacia su interior como si estuviera haciendo presión sobre él; al final, este material adquiriría la forma de la parte de la herramienta que penetra en él. Se supone que el material desalojado de esa región ocuparía otro espacio, y el volumen/área se mantendría constante.

Sea $\partial\Omega$ la interface que define el objeto deformable, representada implícitamente por una función inicial ϕ_0, que estará dada por una función distancia con signo. Suponga que la herramienta define una región (conjunto de puntos) denotada como Γ. Ésta se trasladará conforme lo haga la herramienta. Cuando haya un contacto con el material que se moldea, se espera que ambos objetos compartan una región espacial, esto es, parte del cuerpo de uno debe estar contenido en una parte del otro. Entonces, en el momento de la colisión, los valores de los puntos contenidos dentro de la zona de influencia de la herramienta, serán modificados de acuerdo a un campo de velocidades $\mathbf{V}_H$. Esta velocidad se define en dirección normal a la interface, con orientación hacia el interior del cuerpo deformable y con un módulo igual al de la velocidad v de la herramienta en el momento de colisión, entonces $\mathbf{V}_H = v\frac{\nabla\phi}{||\nabla\phi||}$.

El movimiento de la interface se llevará a cabo de acuerdo a la siguiente ecuación diferencial,

$$\phi_t + (\mathbf{V}_H \cdot \nabla\phi)\chi_\Gamma = \phi_t + v||\nabla\phi||\chi_\Gamma = \phi_t + v\chi_\Gamma = 0. \tag{5.1}$$

donde χ_Γ es la función indicadora correspondiente a la región Γ. Aunque se esté trabajando con una función distancia con signo, se podría emplear el módulo de $||\nabla\phi||$ directamente, sin sustituirlo por la unidad, ya que esta estructura no siempre se conserva con la evolución de la interface, pero también está la opción de la reinicialización, que es útil. Como alternativa a la ecuación diferencial (5.1), está la operación de diferencia mencionada en la **Proposición 2.5.20**. Sin embargo, en este caso también debe

considerarse la representación implícita de la herramienta.

Pues bien, debido a que la interface se mueve en dirección a su interior, su volumen o área interior disminuye. Entonces, se pretende recompensar esta pérdida moviendo el resto de la interface, la parte que queda fuera de la influencia de la herramienta. Una idea es hacerlo en dirección normal así misma (en dirección hacia el exterior) hasta recuperar su volumen (ver Figura 5.1).

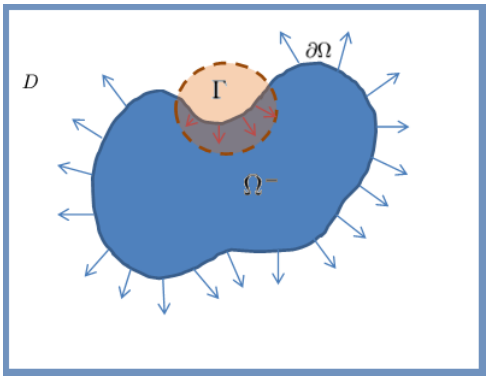


Figura 5.1: Deformación local con conservación de área.

Se define una funcional $Vol : C(\mathbb{R}^n, \mathbb{R}) \longrightarrow \mathbb{R}$ para calcular el volumen/área interior de la interface representada por una función implícita ϕ, haciendo uso de (2.13)

$$Vol(\phi) = \int_{\mathbb{R}^n} (1 - H(\phi))d\mathbf{x},$$

con $n = 2, 3$.

Teniendo la forma de calcular el volumen/área interior de una interface, ahora falta hacer que dicho volumen se conserve durante una deformación (mientras no se separen y ni agreguen partes de material). Una manera es considerar aquella función ϕ que minimice la siguiente funcional:

$$\mathcal{F}(\phi) = \frac{1}{2}(Vol_{\Gamma^c}(\phi) - c)^2, \tag{5.2}$$

donde $c = Vol(\phi_0)$ y $Vol_{\Gamma^c}(\phi) = \int_{\mathbb{R}^n} (1 - H(\phi))\chi_{\Gamma^c} d\mathbf{x}$. La razón de agregar la función característica χ_{Γ^c}, es para asegurar que el volumen de la nueva función sea tomado fuera de la influencia de la herramienta (fuera del conjunto Γ).

Con el fin de encontrar tal función ϕ, se lleva a cabo el procedimiento estándar para funcionales, según la teoría de cálculo de variaciones, equivalente al utilizado para encontrar el valor mínimo de funciones reales, puede consultarse [54]. Esto es, se calcula lo que se conoce como primera variación, una condición necesaria que cumplen los mínimos de una funcional,

$$\frac{d}{d\epsilon}\mathcal{F}(\phi + \epsilon\varphi)\Big|_{\epsilon=0} = 0, \tag{5.3}$$

la cual es suficiente que se cumpla para toda $\varphi \in C^{\infty}(\mathbb{R}^n, \mathbb{R})$.

De modo que

$$\frac{d}{d\epsilon}\ \frac{1}{2}\left(\int_{\mathbb{R}^n}(1-H(\phi+\epsilon\varphi))\chi_{\Gamma^c}d\mathbf{x}-c\right)^2\Bigg|_{\epsilon=0} = ...$$

$$... = \left(\int_{\mathbb{R}^n}(1-H(\phi+\epsilon\varphi))\chi_{\Gamma^c}d\mathbf{x}-c\right)\frac{d}{d\epsilon}\int_{\mathbb{R}^n}(1-H(\phi+\epsilon\varphi))\chi_{\Gamma^c}d\mathbf{x}\Bigg|_{\epsilon=0} = ...$$

$$... = (Vol_{\Gamma^c}(\phi+\epsilon\varphi)-c)\int_{\mathbb{R}^n}(-H'(\phi+\epsilon\varphi))\chi_{\Gamma^c}\varphi d\mathbf{x}\Bigg|_{\epsilon=0} = -(Vol_{\Gamma^c}(\phi)-c)\int_{\mathbb{R}^n}H'(\phi)\chi_{\Gamma^c}\varphi d\mathbf{x} = ...$$

$$... = \int_{\mathbb{R}^n}(-(Vol_{\Gamma^c}(\phi)-c)H'(\phi)\chi_{\Gamma^c}\varphi)d\mathbf{x} = 0, \tag{5.4}$$

para toda $\varphi \in C^{\infty}(\mathbb{R}^n, \mathbb{R})$. Cabe aclarar que la justificación de derivar la función de Heaviside H en el procedimiento anterior es debido a que se está tomando en realidad como una distribución en lugar de una función en sí misma, por lo que es derivable en cero y su derivada es conocida como *función delta*[1] ($\delta(\phi)$).

Por tanto, según el *Teorema fundamental del cálculo variacional*[49],

$$-(Vol_{\Gamma^c}(\phi)-c)\delta(\phi)\chi_{\Gamma^c} = 0. \tag{5.5}$$

Entonces, se introduce una variable de tiempo artificial t como $\phi(\mathbf{x}, t))$, para aplicar un método de optimización numérica como es el *gradiente descendente*, tal que el valor inicial de ϕ es $\phi(\mathbf{x}, 0) = \phi_0(\mathbf{x})$, queda como

$$\phi^{n+1} - \phi^n = -\Delta t(Vol_{\Gamma^c}(\phi)-c)\delta(\phi)\chi_{\Gamma^c}, \tag{5.6}$$

donde Δt es el paso de tiempo y $\phi^n = \phi(\mathbf{x}, t_n)$, como se mostró en el capítulo anterior. Si la parte derecha de esta esta ecuación converge a cero, entonces se alcanza la función ϕ que minimiza (5.2), es decir permite que el volumen/área del objeto deformable no varíe.

Para la implementación de (5.6), se utiliza una aproximación númerica de la función δ diferenciable en todo el dominio (ver Figura 5.2)

$$\delta_\epsilon(\phi) = \begin{cases} 0, & si\ \phi < -\epsilon, \\ \frac{1}{2\epsilon} + \frac{1}{2\epsilon}cos(\frac{\pi\phi}{\epsilon}), & si\ -\epsilon \leqslant \phi \leqslant \epsilon, \\ 0, & si\ \phi > \epsilon. \end{cases} \tag{5.7}$$

el valor de ϵ puede escogerse según convenga y puede variar, aunque un valor mínimo recomendado es $1.5\Delta x$ [49], esto es, debe ser más grande que el tamaño de paso de la malla uniforme utilizada en la implementación, con el fin de afectar a los conjuntos de nivel de la función ϕ más cercanos a la interface.

[1] En la teoría de distribuciones, la derivada de la función de Heaviside H, H', es la función delta, δ, definida en todo los números reales tal como $\delta(x) = 0$, si $x \neq 0$, y que cumple $\int_{\mathbb{R}}\delta(x) = 1$, por lo que su valor en cero se podría definir como infinito. Puede consultar [55] para más información sobre la función δ y la *teoría de distribuciones*.

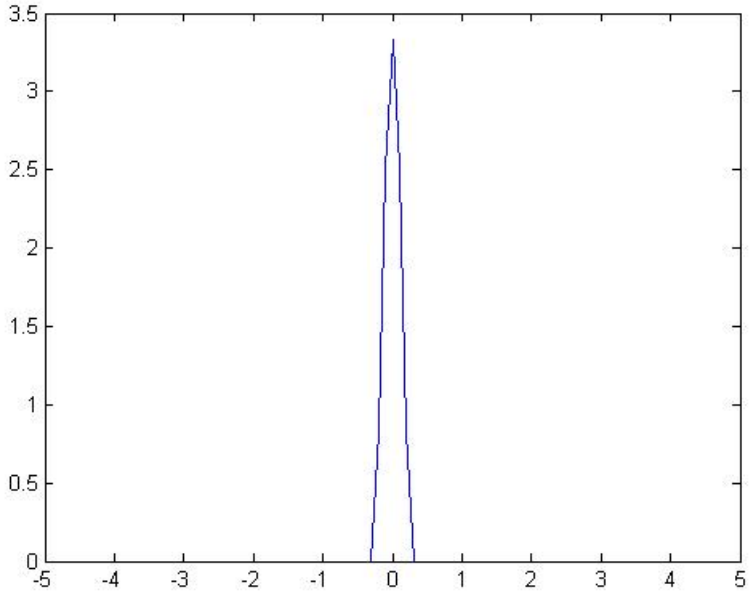


Figura 5.2: $\delta_\epsilon(x)$, aproximación derivable de la función delta original, $\delta(x)$.

Al considerar otros conjuntos de nivel además del correspondiente a la interface, y si tomamos en cuenta que utilizar la función distancia con signo para su representación, la norma de $||\nabla\phi|| = 1$ aparece implícitamente en (5.6), lo que hace que esta ecuación sea la semidiscretización (discreta en la variable temporal, pero continua en las variables espaciales) de una ecuación de movimiento como se vió en el capítulo anterior, propiciando el desplazamiento de la interface al hacer evolucionar los conjuntos de nivel que se encuentran alrededor de ella. Se pueden considerar todos y no solamente los conjuntos que están cercanos al conjunto de nivel cero, que es como lo delimita la función δ, al eliminar esta función. Con esto se tendría

$$\phi^{n+1} = \phi^n - \Delta t(Vol_{\Gamma^c}(\phi) - c)||\nabla\phi||\chi_{\Gamma^c}, \tag{5.8}$$

que como puede observarse, es la discretización explícita en tiempo de la ecuación de movimiento

$$\phi_t = (Vol_{\Gamma^c}(\phi) - c)||\nabla\phi||\chi_{\Gamma^c}. \tag{5.9}$$

Las condiciones de frontera para el modelo no salen de forma natural del procedimiento variacional llevado a cabo en (5.4), por lo que se decidió utilizar artificialmente una condición de frontera conocida como *adiabática*. En física, el significado de un proceso adiabático en un sistema, es que dicho sistema (ya sea un cable o, en este caso, el dominio D) está aislado y no hay un flujo de energía, calor o materia su frontera al exterior. Matemáticamente hablando esto queda establecido por $\frac{\partial\phi}{\partial\mathbf{n}} = 0$, donde $\mathbf{n} = (n_1, n_2)$ es el vector normal a la frontera del dominio D, pero como existe $\nabla\phi$ en esos puntos, entonces

$$\nabla\phi \cdot \mathbf{n} = \frac{\partial\phi}{\partial\mathbf{n}} = 0, \tag{5.10}$$

pero ya que el dominio D es un rectángulo cuyos lados son paralelos a los ejes coordenados, sus normales también serán paralelos a los ejes coordenados, por lo que una de sus coordenadas será cero y la otra no, entonces la relación $\frac{\partial \phi}{\partial x} n_1 + \frac{\partial \phi}{\partial y} n_2$ generada por la ecuación (5.10) hace que las parciales de x del gradiente correspondientes a los lados verticales de D se anulen, mientras que en los lados horizontales se anulan las parciales con respecto a y. Las integrales de volumen/superficie son discretizadas puntulamente

Habiendo establecido todo lo anterior, lo siguiente es la implementación de la deformación local y el movimiento de compensación de volumen, suponiendo que ya ha colisionado una herramienta virtual con un material deformable, y la parte de la herramienta que queda dentro del material deformable empieza a ejercer su influencia a través de la ecuación (5.1) con un módulo de velocidad v igual al módulo de la velocidad de la herramienta, mientras que en el resto de la frontera del material se expande en dirección normal a su frontera, todo esto queda plasmado en el **Algoritmo 2**.

Sólo resta presentar los experimentos llevados a cabo para probar el modelo y la renderización háptica, que es en lo que consistirá el último capítulo.

Algoritmo 2 Deformación local y compensación de volumen

Entrada: Los valores de función ϕ que representa al material deformable y los puntos P_H de la representación explícita de la herramienta virtual y su velocidad v. ONBSERVACIÓN: Se supone que la herramienta y el material deformable están en colisión.

Salida: Los valores de la representación implícita de la nueva interface.

1: Se calcula el volumen inicial de ϕ, $c = Vol(\phi)$.
2: Se detectan los puntos de la malla que están dentro de la zona de influencia Γ de la herramienta virtual.
3: Se haya el volumen $Vol_{\Gamma^c}(\phi)$ del objeto deformable que se encuentra fuera de la zona de influencia de la herramienta y también se encuentra la diferencia $dif = Vol_{\Gamma^c}(\phi) - c$.
4: Se calcula el gradiente $\nabla\phi$, tomando condiciones de frontera adiabáticas.
5: Se llama a la función de renderización háptica para utilizar la sección de colisión, que da como resultado un valor a la variable *colision*.
6: **mientras** $(colision \neq 0)||(dif \neq 0)$ **hacer**
7: **si** $colision \neq 0$ **entonces**
8: Se aplica la ecuación (5.1) utilizando la norma del gradiente $\nabla\phi$, la zona de influencia de la herramienta y la velocidas de deformación v de la herramienta para realizar la deformación local.
9: **fin si**
10: **si** $dif \neq 0$ **entonces**
11: Se encuentra el complemento Γ^c de la zona de influencia de la herramienta.
12: Se aplica la ecuación (5.6) ó (5.8) con el complemento de la zona de influencia de la herramienta, la diferencia dif, y según sea el caso la función delta δ o la norma del gradiente $\nabla\phi$.
13: Se calcula el nuevo volumen $Vol_{\Gamma^c}(\phi)$ del objeto deformable y tambien la diferencia entre el volumen original y el actual $dif = Vol_{\Gamma^c}(\phi) - c$.
14: **fin si**
15: Opcional: Se aplica reinicialización y se elige su frecuencia.
16: **fin mientras**
17: **devolver** los nuevos valores de la función ϕ

Capítulo 6

Resultados experimentales y conclusiones

Este último capítulo es la culminación de la tesis, en donde se muestran diferentes experimentos para saber cómo funcionan los algoritmos de renderización háptica, el modelo de deformación local y la compensación de volumen, así como la combinación de los tres. Se toman en cuenta algunas formas básicas bidimensionales tanto para la herramienta como para el material deformable, en algunos de esos experimentos también se considera el tiempo en que tardó en ejecutarse el programa y sus componentes, así como el tamaño de paso espacial y temporal, y se analizan las implicaciones parara valorizar tales algoritmos y modelos.

6.1. Renderización háptica

A lo largo de esta sección se muestran los experimentos que involucran la parte de detección de colisión y cálculo de fuerzas de la renderización háptica. Se comparan aquellos resultados calculados con $\mathbf{F} = \sum_i \mathbf{F}_i$ y $\mathbf{F} = \frac{1}{n}\sum_i \mathbf{F}_i$, además de otros factores.

Se utiliza como herramienta una circunferencia con centro inicial localizado en (50,73) y de radio 15 unidades, y como material deformable un rectángulo con centro en (50,20), de base 50 unidades y de altura 30 unidades. La herramienta virtual se desplaza en línea recta hacia el centro del rectángulo a una velocidad de 7 unidades espaciales por unidad de tiempo, hasta colisionar con el rectángulo. Primero se comparó la manera en que afecta a la fuerza final, aquella que se aplica al centro de masa de la herramienta, la ecuación que se utiliza para calcularla, esto es, si se utiliza $\mathbf{F} = \sum_i \mathbf{F}_i$ o $\mathbf{F} = \frac{1}{n}\sum_i \mathbf{F}_i$. Se tiene en cuenta que el índice de rigidez $\kappa = 1$, y el número de puntos discretos de la herramienta es 50 (Figura 6.1 y Figura 6.2).

En la visualización, la herramienta está formada por cruces pequeñas que forman su contorno y

delimitan el interior, a diferencia, el objeto del ambiente con el que tiene colisión está dibujado por una línea continua. La fuerza se muestra como una línea que está localizada en el centro de la herramienta.

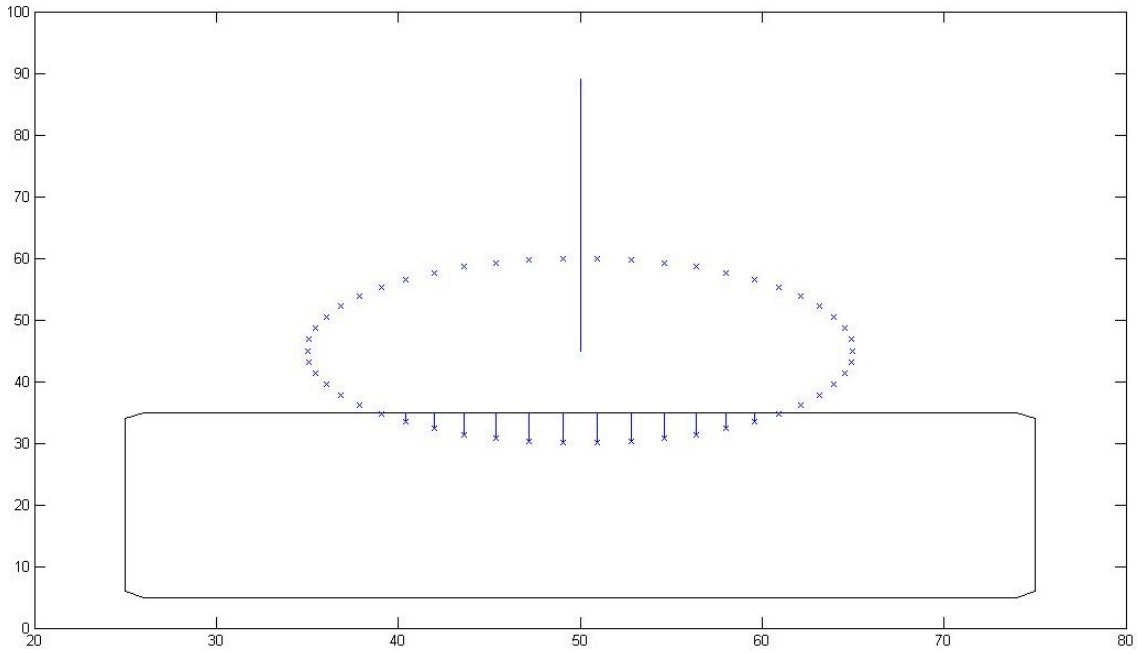


Figura 6.1: $\mathbf{F}$ se calculó utilizando $\mathbf{F} = \sum_i \mathbf{F}_i$

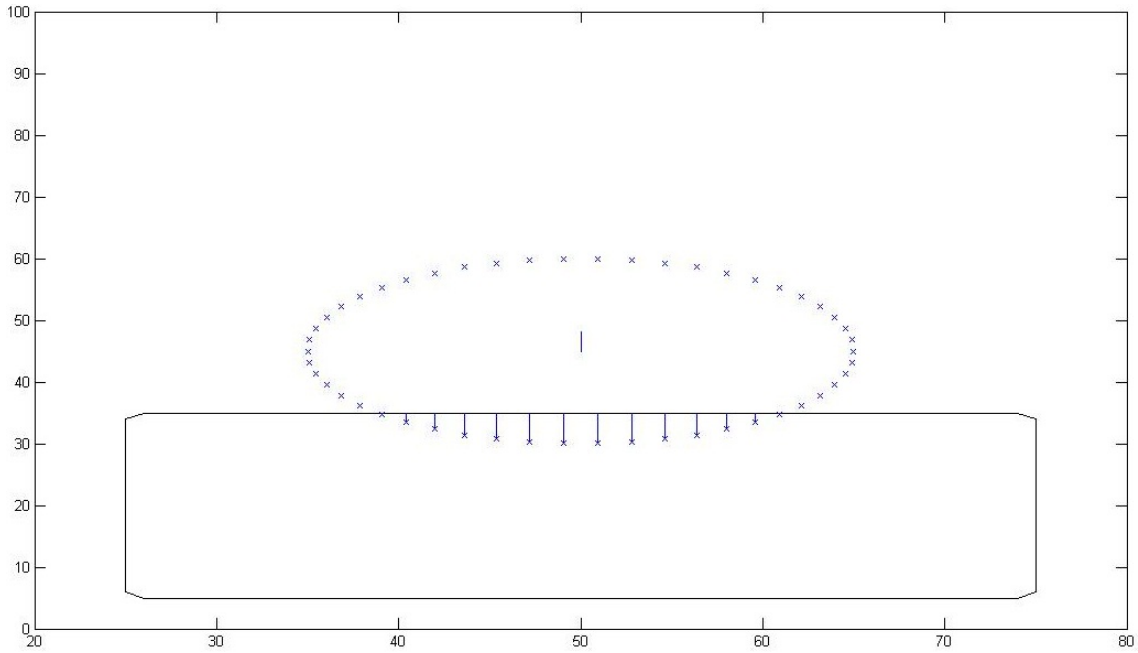


Figura 6.2: $\mathbf{F}$ se calculó utilizando $\mathbf{F} = \frac{1}{n} \sum_i \mathbf{F}_i$

Puede observarse que en el primer caso, cuando se utiliza la suma de fuerzas solamente, la fuerza

resultante es bastante grande (44 unidades de fuErza) comparado con la fuerza promedio del segundo (3.147 unidades de fuerza), por lo que el primer modelo sería propenso a crear inestabilidad en la renderización háptica. También, las magnitudes de las sumas de fuerza varía bastante según cuántos puntos de la herramienta se utilice en su discretización. En las Figuras 6.3 y 6.4 se muestran los resultados cuando la herramienta está formada por 25 puntos; las Figuras 6.5 y 6.6 muestran los resultados cuando la herramienta cuenta con 100 puntos.

	Número de puntos en colisión	Magnitud e la fuerza $\mathbf{F}$	Tiempo total
Figura 6.3	7	22 unds.	0.4 s.
Figura 6.4	7	3.14 unds.	0.4 s.

El valor de la fuerza con $\frac{1}{n}\sum_i \mathbf{F}_i$, se mantiene el mismo que cuando la herramienta tenía 50 puntos discretizados, mientras que en el otro caso hay un cambio de muy brusco, una diferencia de 22 unidades.

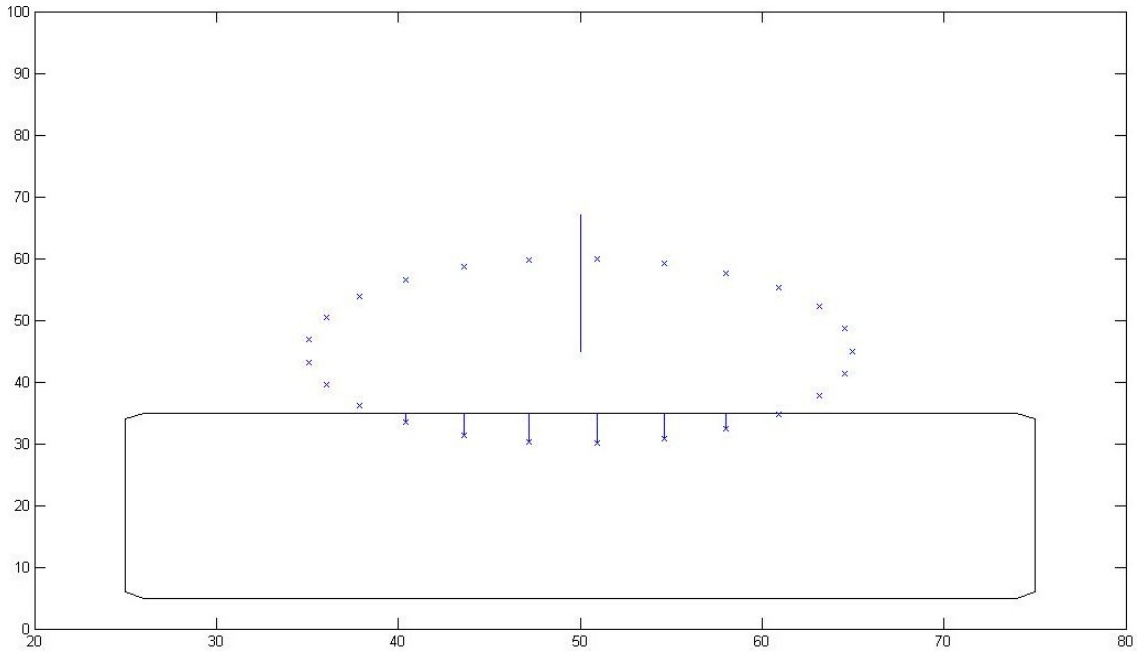


Figura 6.3: Con 25 puntos y $\mathbf{F} = \sum_i \mathbf{F}_i$

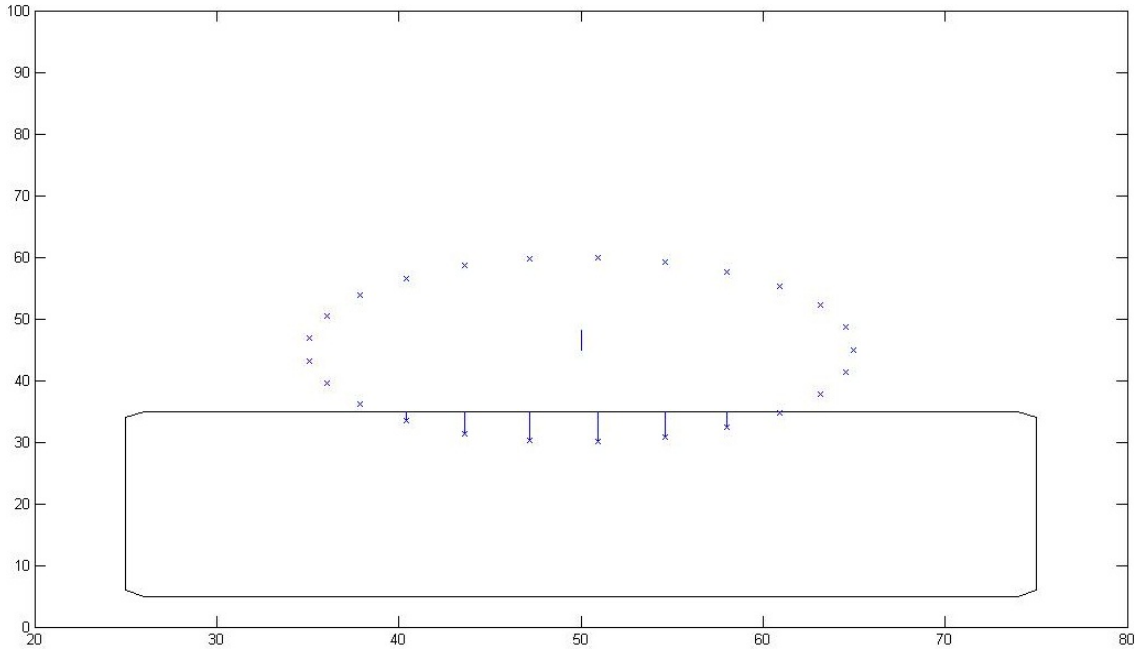


Figura 6.4: Con 25 puntos y $\mathbf{F} = \frac{1}{n}\sum_i \mathbf{F}_i$

	Número de puntos en colisión	Magnitud e la fuerza **F**	Tiempo total
Figura 6.5	27	88 unds.	0.5 s.
Figura 6.6	27	3.26 unds.	0.5 s.

En la Figura 6.5 se dispara el tamaño de la fuerza resultante, de tal manera que sale del recuadro de la gráfica. El valor de la fuerza correspondiente a la figura 6.6 se mantiene coherente con el valor de la magnitud cuando el número de puntos eran 25 y 50. El modelo del promedio de fuerzas parece ser el indicado hasta ahora para calcular el valor de la fuerza.

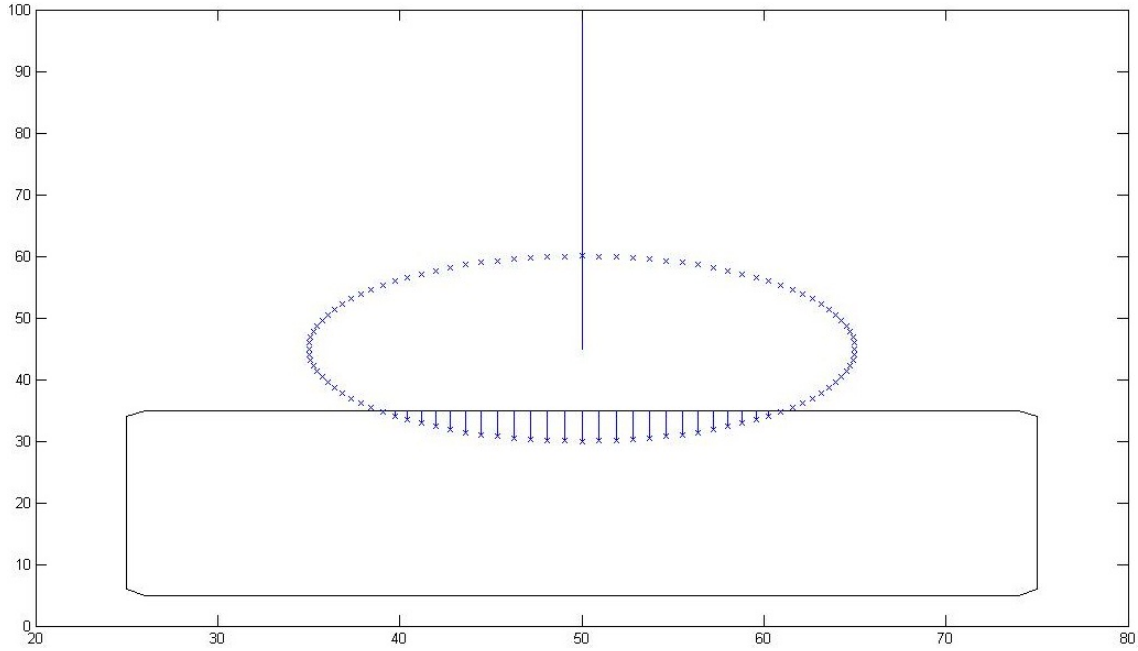


Figura 6.5: Con 100 puntos y $\mathbf{F} = \sum_i \mathbf{F}_i$

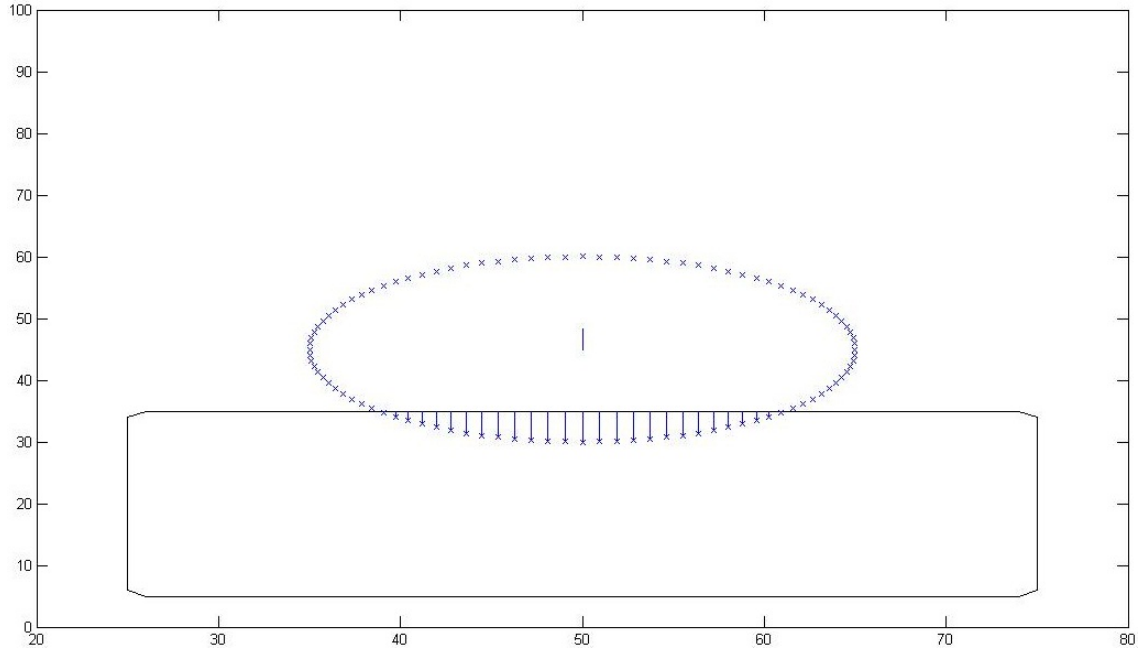


Figura 6.6: Con 100 puntos y $\mathbf{F} = \frac{1}{n} \sum_i \mathbf{F}_i$

Pareciera que la norma de la velocidad de la herramienta tiene una relación directa con el tamaño de la fuerza calculada, pero esto no es necesariamente cierto, ya que también depende a qué distancia se encuentren del objeto con el que va a tener colisión. El otro factor de la velocidad de la herramienta, su dirección, parece no ser que en algunos casos resulta ser inverosímil con la fuerza resultante encontrada.

En el siguiente ejemplo se pone de manifiesto tal aseveración, en ambos casos las herramientas se acercaron hacia la punta superior izquierda del rectángulo desde direcciones diferentes; en la Figura 6.7 el movimiento fue de manera diagonal y en la Figura 6.8 de forma vertical, pero ambos tienen una dirección en la fuerza resultante,en donde el primero sí parece reflejar la dirección de la velocidad y el contacto, mientras que en el segundo hay una discontinuidad y un cambio brusco en la dirección.

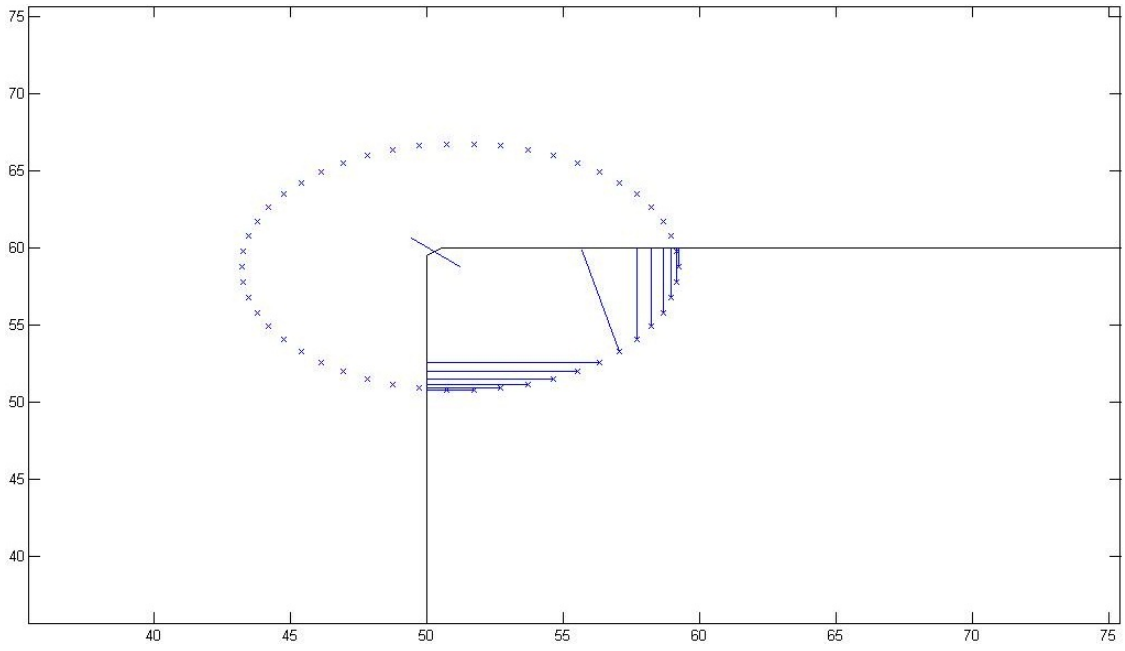


Figura 6.7: La dirección de la herramienta fue diagonal desde el extremo superior izquierdo

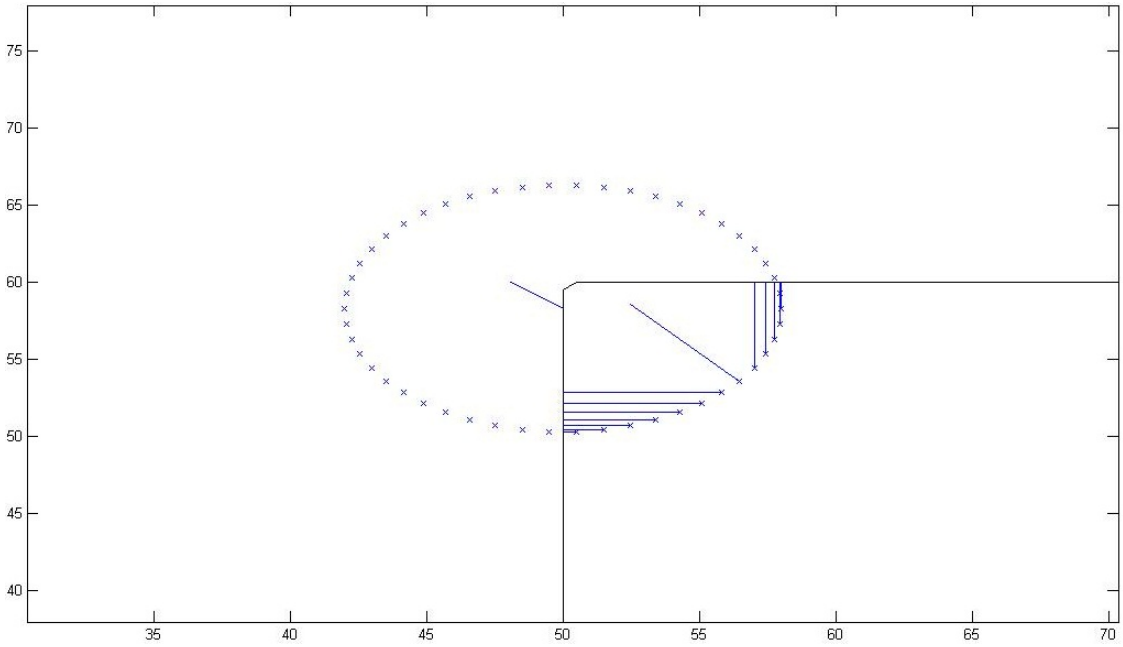


Figura 6.8: La dirección de la herramienta fue vertical desde arriba.

Otro factor importante en la magnitud de la fuerza es el índice de rigidez κ, en todos los resultados previos se ha manejado $\kappa = 1$. Claramente este factor cambia proporcionalmente con la magnitud de la fuerza resultante como se puede observar comparando las figuras 6.1 y 6.2 de esta sección con las siguientes. κ varía con los valores 0.5 y 1.5.

	Número de puntos en colisión	Magnitud e la fuerza **F**	Tiempo total
Figura 6.9	14	22 unds.	8.4 s.
Figura 6.10	14	1.6 unds.	8 s.

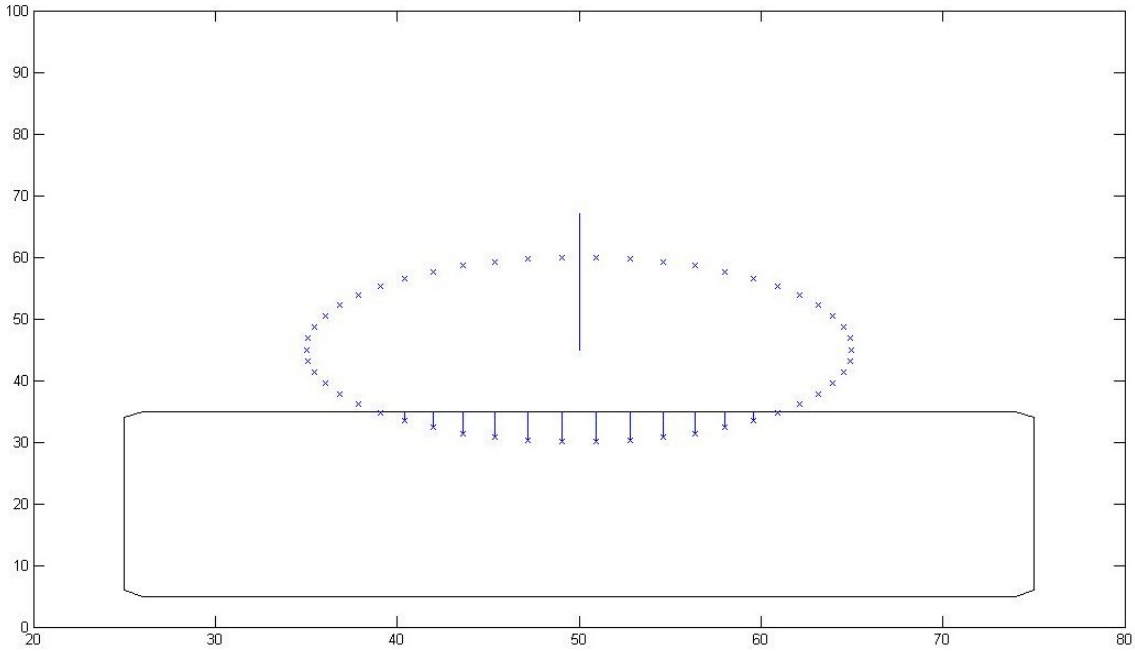


Figura 6.9: κ =0.5 con $\mathbf{F} = \sum_i \mathbf{F}_i$.

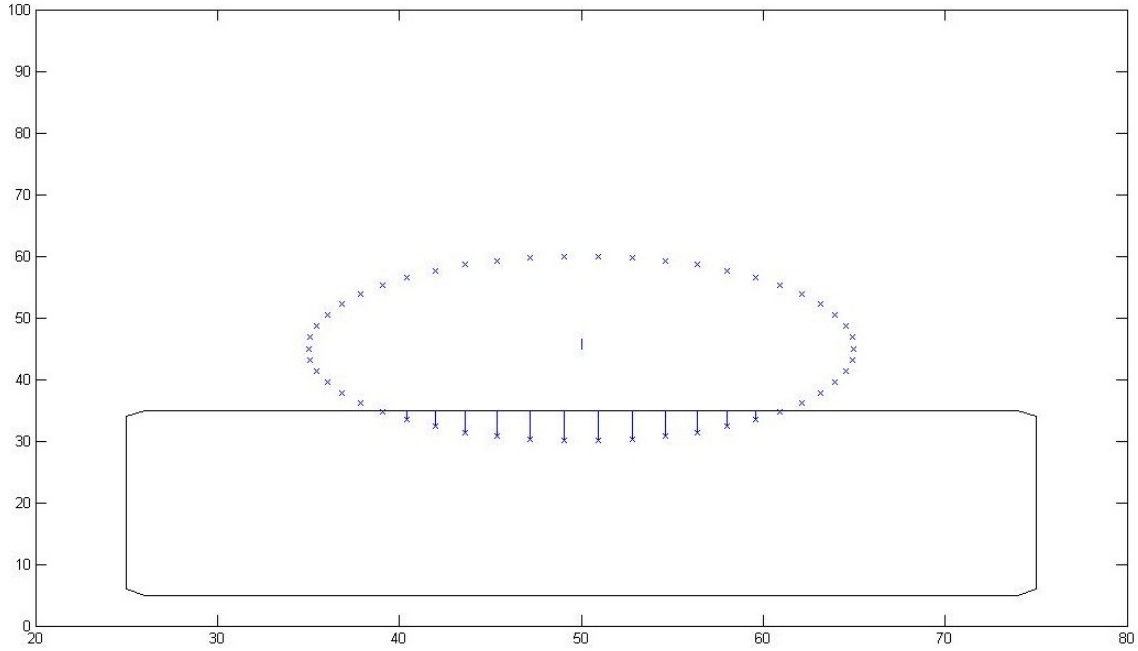


Figura 6.10: κ =0.5 con $\mathbf{F} = \frac{1}{n} \sum_i \mathbf{F}_i$.

	Número de puntos en colisión	Magnitud e la fuerza $\mathbf{F}$	Tiempo total
Figura 6.11	14	66 unds.	0.7 s.
Figura 6.12	14	5 unds.	0.4 s.

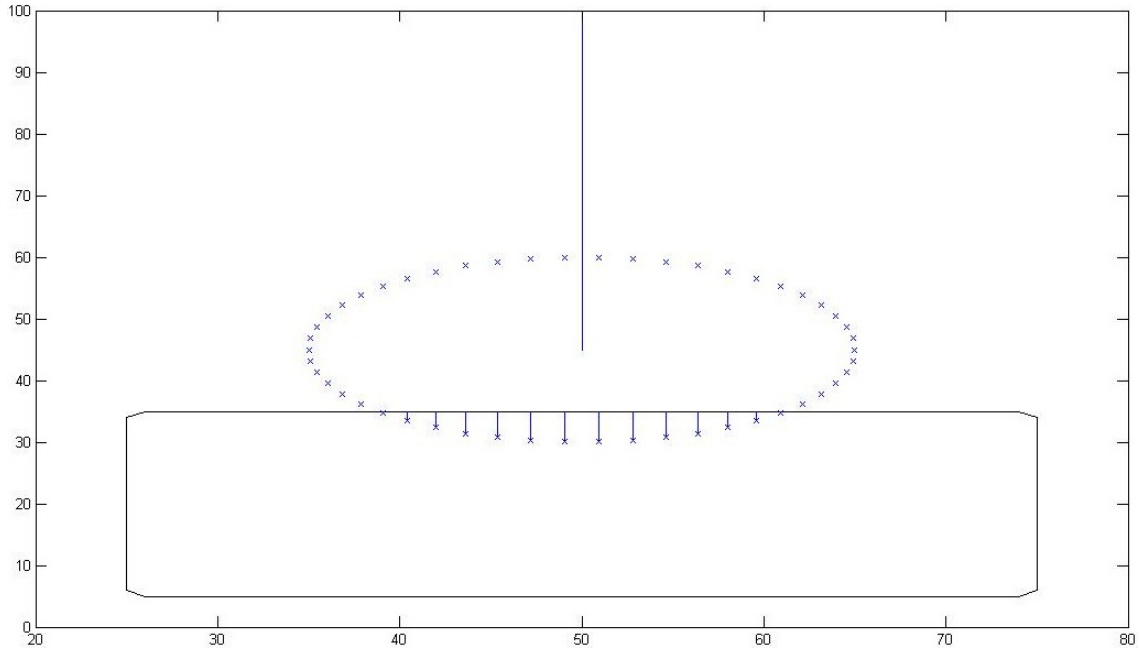


Figura 6.11: κ =1.5 con $\mathbf{F} = \sum_i \mathbf{F}_i$.

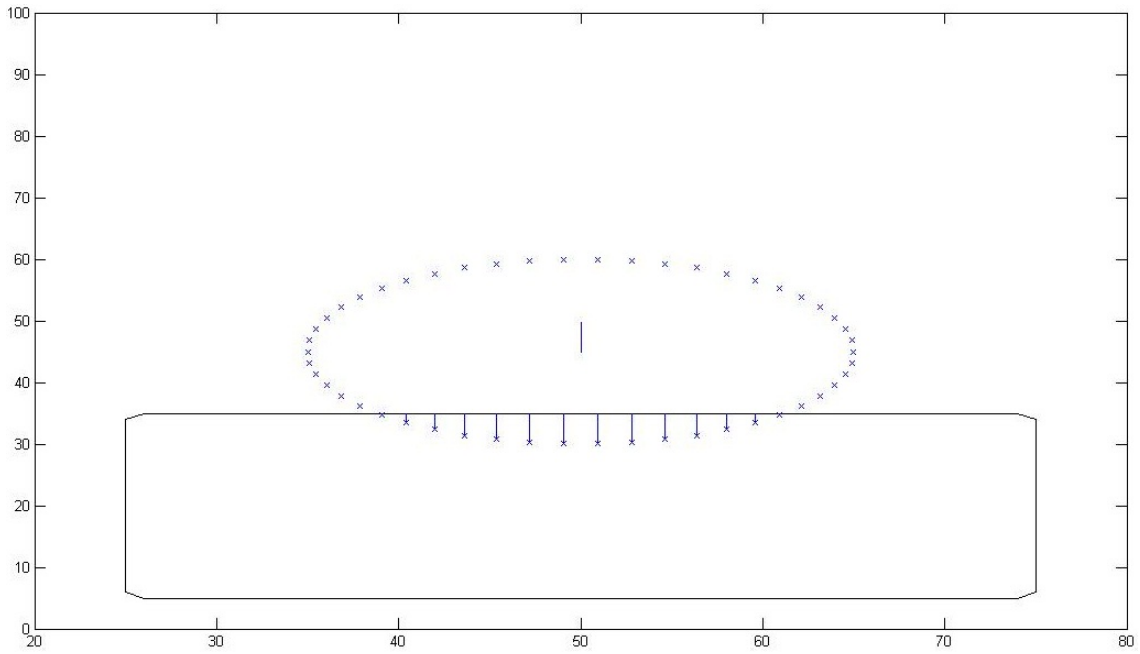
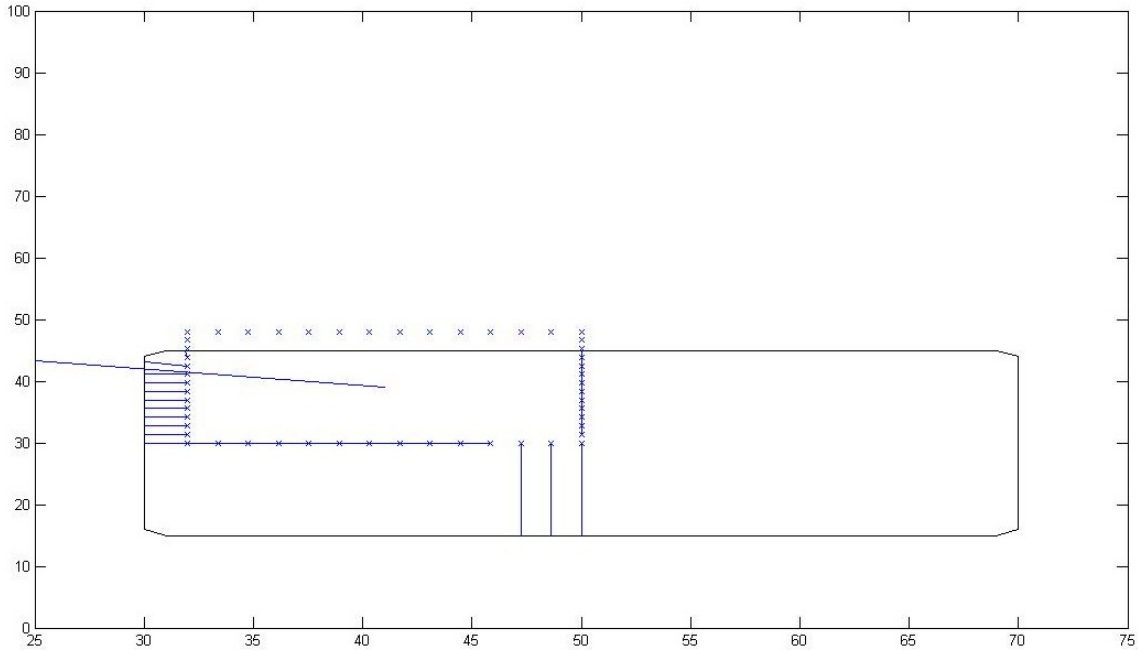
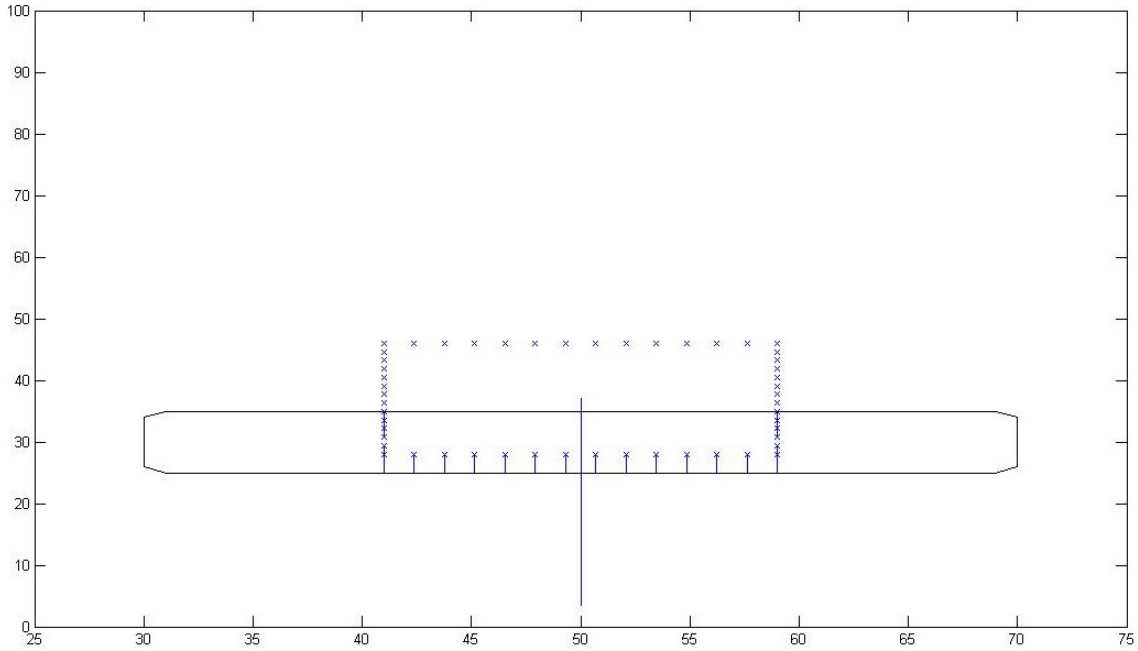


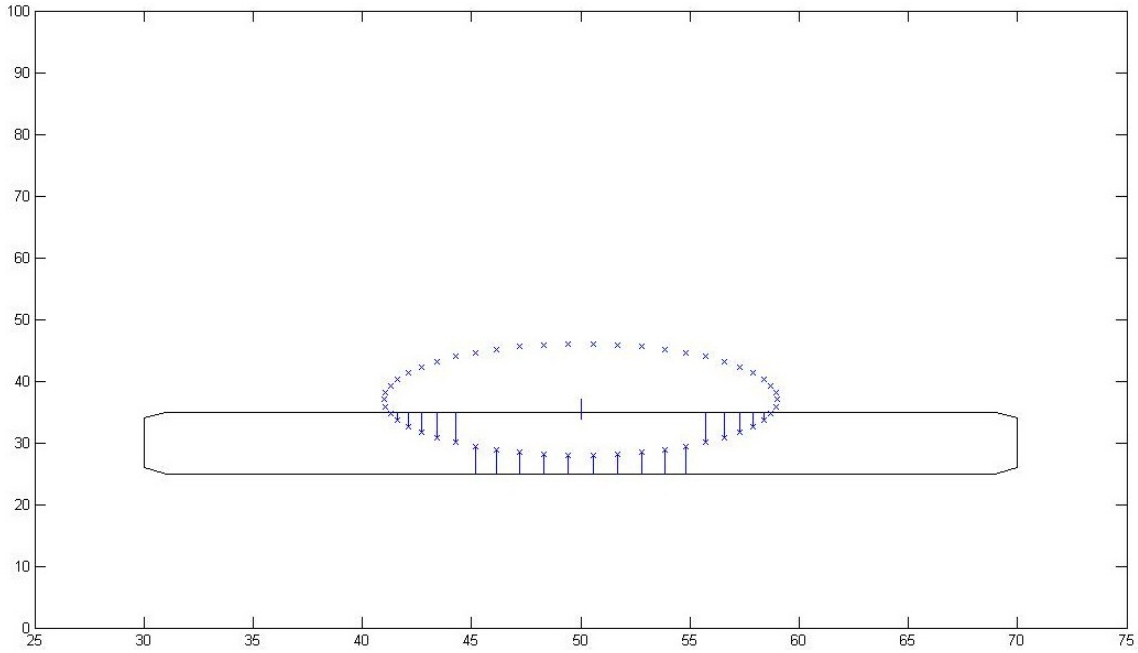
Figura 6.12: κ =1.5 con $\mathbf{F} = \frac{1}{n}\sum_i \mathbf{F}_i$.

En lo siguiente se darán ejemplos de discontinuidades en la fuerza en la dirección de la fuerza. Anteriormente, en la figura se muestr una discontinuidad en la dirección, pero no es necesario tocar a una punto para que haya alguna. Cada herramienta se acercó de manera vertical hacia el objeto ambiente y solamente el primer caso se acerca a una punta. Notar que muchos puntos de la cara horizontal que

está dentro de la herramienta tienen fuerzas que son paralelas a la cara de donde vino el contacto, lo que es evidentemente una discontinuidad de dirección de fuerza un tanto particular. También este caso ilustra una idea de lo que no debería o debería suceder, los únicos puntos que se deberían considerar para calcuar las fuerzas de contacto deberían ser los puntos de la superficie de la herramienta que tuvieron un primer acercamiento con el objeto.







El modelo en general trabaja bien, y a pesar de ser hasta cierto punto una adaptación simple, la función distancia con signo es de bastante ayuda para calcular las magnitudes y las direcciones de las fuerzas que actúan en los puntos de la herramienta con el *método de penalización*, que es el utilizado en este trabajo. Los inconvenientes que se destacan, como los cambios bruscos y discontinuidades en las fuerzas, son habituales en todas los métodos de renderización háptica en un primer acercamiento. Para trabajos futuros se podría implementar en un sistema háptico completo, donde se analice la estabilidad del sistema, así como implementar una mejora en la detección de colisión y en elcálculo de fuerzas. También se podría adaptar a más grados de libertad en un espacio tridimnensional

6.2. Deformación local

La deformación local consiste básicamente en definir la función indicadora χ_Γ de la región que ocupa la herramienta Γ, de forma que, a partir de la representación explícita de ésta se construye una nueva malla, como se hizo para definir a la función ϕ, sólo que en este caso los valores serán los valores de dicha función χ_Γ.

Debido a que las formas de la herramienta consideradas son sencillas, se pueden utilizar las representaciones explícitas directamente. Por ejemplo, en el caso de ser un círculo de radio r con centro en (x_0,y_0), Γ contiene aquellos puntos (X, Y) que cumplen la desigualdad

$$\sqrt{(x_0 - X)^2 + (y_0 - Y)^2} \leqslant r.$$

Cuando la herramienta es un cuadrado, cuya base está alineada con respecto al eje x, con centro en (x_0,y_0) y con lados de longitud l, entonces los puntos (X,Y) en Γ cumplen las siguientes dos condiciones:

$x_0 - l/2 \leqslant X \leqslant x_0 + l/2$, $y_0 - l/2 \leqslant Y \leqslant y_0 + l/2$. Sin embargo, se puede utilizar o crear un programa más general que convierta a una forma explícita en immplícita y trabajar con ella sin ningún problema, simplemente habría que adaptarLO en el algoritmo general.

La implementación de la ecuación (5.1) con condiciones de fontera adiabática, se llevó a cabo utilizando diferencias finitas hacia delante en la derivada parcial temporal y diferencias centrales en las derivadas parciales espaciales.

En los experimentos se tomó en cuenta el valor de condición CFL de dos dimensiones $\alpha = \Delta t \frac{max\{||\mathbf{V}_H||\}}{h^2}$[1], así como también la reinicialización. En el caso de los parámetros de los que depende α, el que se considera es el valor de paso de tiempo Δt, los otros dos casos, tanto el módulo de la velocidad $\mathbf{V}_H$ y el del paso espacial h se pueden ver como un reflejo de lo que pasa al variar Δt. Además, el tamaño de paso espacial h implica mayor complejidad y costo computacional para resolver la ecuaciónn (5.5) y por tanto la deformación. Los valores de Δt que se consideran son 1, 0.1 y 0.01.

Se fija el módulo de la velocidad de deformación de la herramienta en 7 unidades espaciales sobre unidad de tiempo, el número de puntos en cada eje coordenado de la malla es $n = 201$, de tal forma que el tamaño de paso espacial en ambos ejes coordenados es $h = 0.5$. Las herramientas consideradas tienen como dimensiones: el círculo es de radio 9 unidades y las longitudes de las caras de los cuadrados son también de 9 unidades; el centro de masa de cada uno es (65,63). Los materiales u objetos que se deforman tienen como dimensiones: tanto el radio del círculo como la longitud del lado del cuadrado son de 18 unidades; sus centros están localizados en (50,50).

En las figuras que siguen se muestran los resultados de los experimentos, las herramientas están definidas visualmente con cruces en su contorno como se hizo en la sección anterior, los objetos que se deforman están definidos por líneas sólidas (negro para la interface final y verde para la interface inicial). En algunas figuras se muestran además una banda de líneas a color, que son otros conjuntos de nivel alrededor de la interface (sus valores van desde -3 a 3, pasando de 0.5 en 0.5), que es el conjunto de nivel cero, así que la interface se encuentra en medio de las líneas azul claro y las amarillas.

Las tablas que acompañan a las figuras muestran datos significativos de cada ejecución o experimento como el tiempo total, el número de iteraciones de evolución de interface, el tiempo que le lleva realizar la reinicialización y el número de reinicializaciones que se llevaron a cabo. Los primeros resultados que se muestran es para el caso cuando el tamaño de paso de tiempo es $\Delta t = 1$, las formas de ambos objetos se restringirán a círculos para mostrar los resultados en cada uno de los experimentos como se ve en seguida.

Como se puede observar de las gráficas, el resultado de deformación local es prácticamente el mismo utilizando la reinicialización de la interface o no, pero hay una diferencia de tiempo significativa al considerarla, sobre todo si se lleva la mayor pate de tiempo de ejecución como se muestra en la tabla que sigue de las Figuras 6.13 y 6.14. Lo apropiado sería no utilzarla tan frecuentemente, como se hizo en este caso que se aplico en cada reinicialización, para que no ocupe tanto tiempo y sirva también en los casos

[1] Recordar que $\mathbf{V}_H$ es la velocidad con la que la herramienta deforma a la interface y que $h = min\{\Delta x, \Delta y\}$, pero aquí se trabja con $\Delta x = \Delta y = h$.

en que sea conveniente llevarla a cabo.

Una observación se puede realizar al checar las figuras de los conjuntos de nivel. Como se dijo anteriormente, se graficaron los que tienen valores de -3 a 3, saltando de 0.5 en 0.5 entre una y otra. Sin embargo en el resultado final, en la zona de la herramienta, no se puede ver ninguna línea de la banda de colores, además de la interface inicial de color verde, eso es porque se encuentran totalmente pegadas al borde de la herramienta junto al conjunto de nivel cero, pero no se pueden ver. Dentro de la herramienta todavía siguen habiendo conjuntos de nivel, simplemente son de valores mayores que no se consideran en las gráficas.

	No. de reinicializaciones	Tiempo de reinicialización	Iteraciones	Tiempo total
Figura 6.13	0	0	4	75.6 s.
Figura 6.14	4	118.8 s.	4	164.3 s.

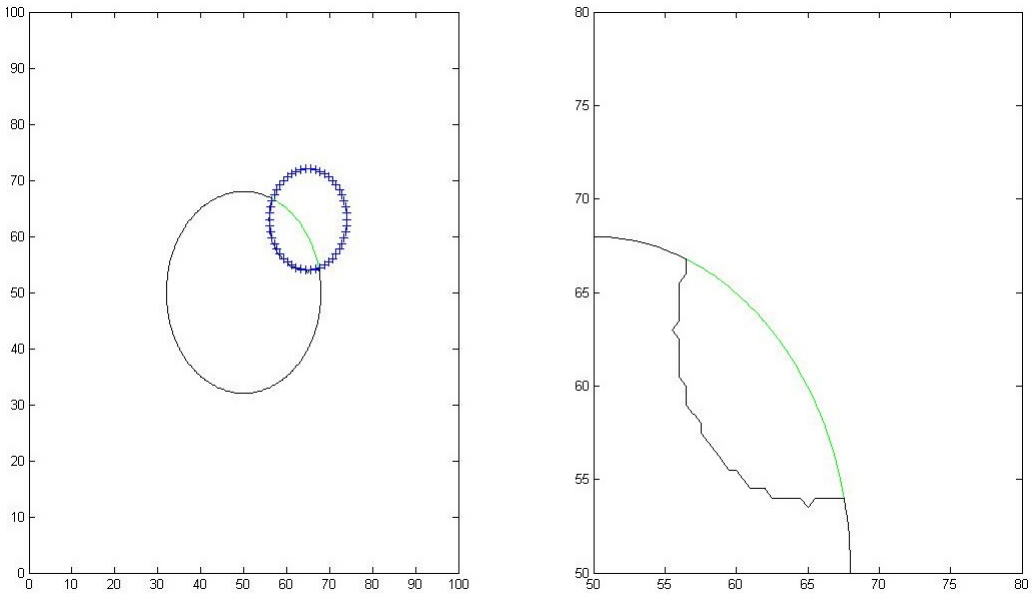


Figura 6.13: $\Delta t = 1$ y sin reinicialización.

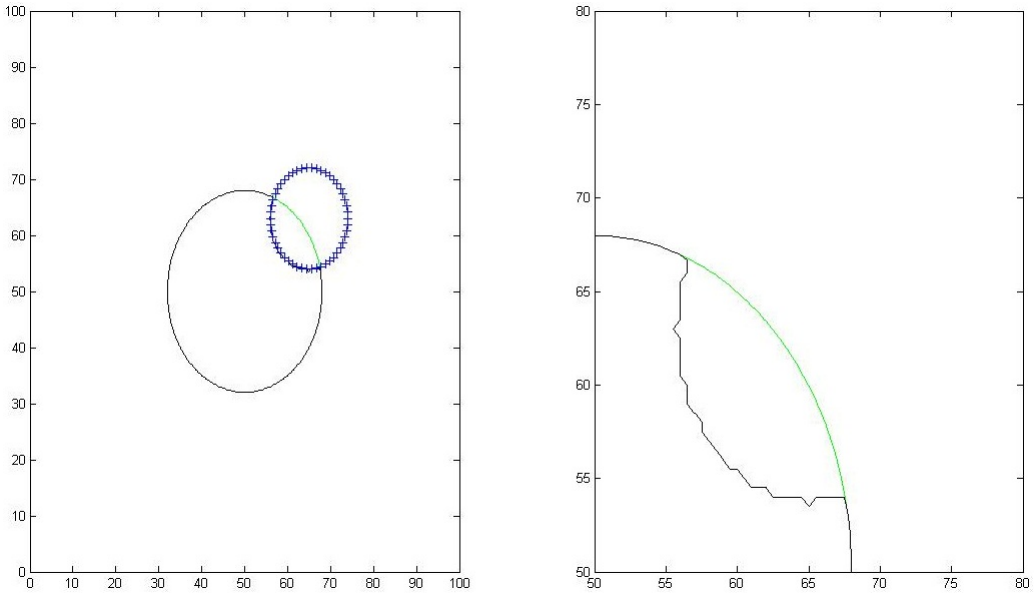


Figura 6.14: $\Delta t = 1$, con reinicialización en cada iteración.

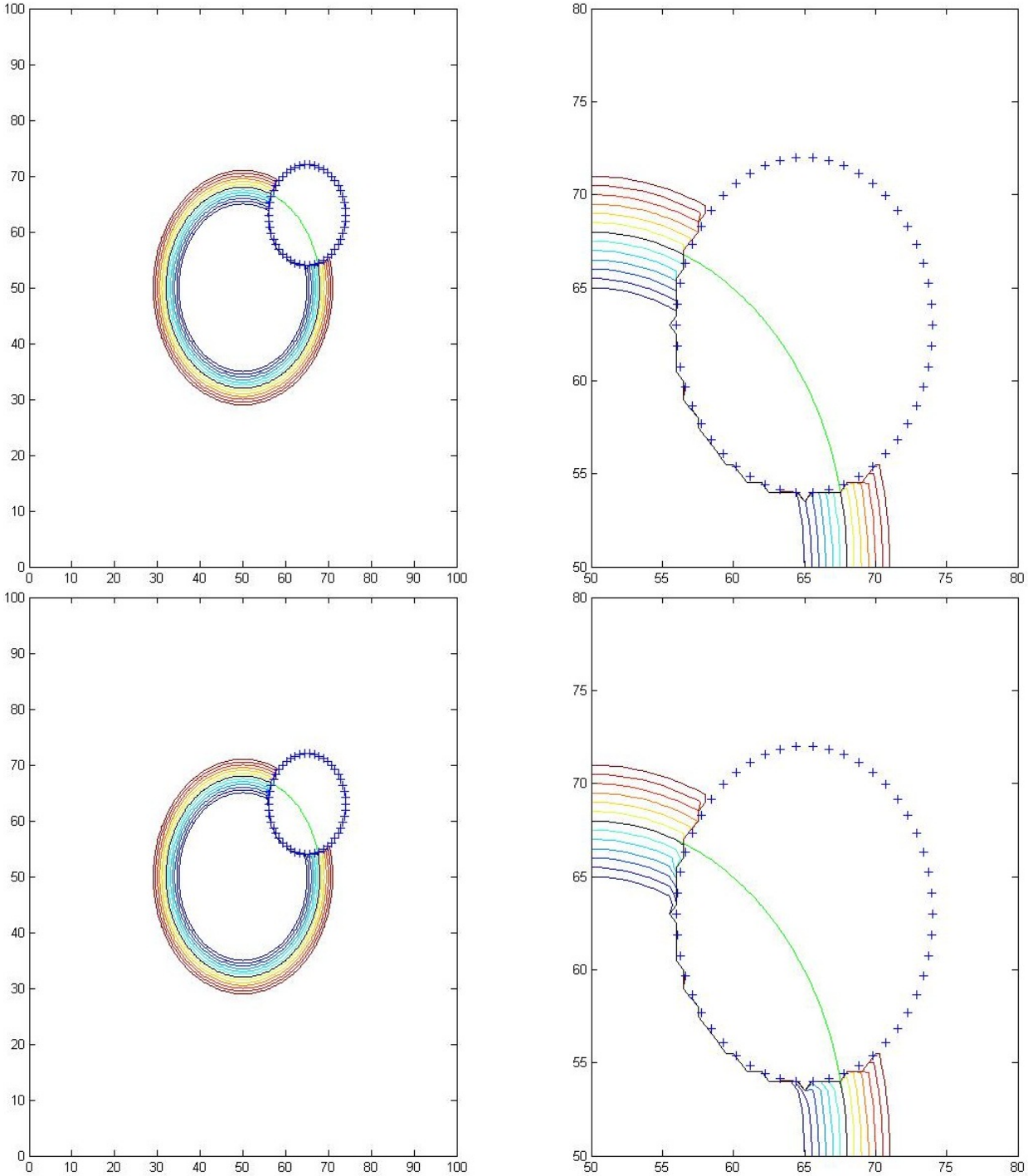


Figura 6.15: (*arriba*) sin reinicialización.(*abajo*) con reinicialización en cada iteración.

Para los casos en que $\Delta t = 0.1$ y 0.01, en la parte visual las interfaces finales son prácticamente iguales a los anteriores, tampoco hay mucha diferencia en los conjuntos de nivel, excepto que en la Figura 6.21 hay algunos residuos de los conjuntos de nivel que se consideran en la graficación, lo que muestra que no hubo un movimiento estable de los conjuntos de nivel. El tiempo total aumenta mientras Δt es

más pequeño, y todavía más con la reinicialización. Así que en la deformación local, no es conveniente utilizar las reinicializaciones, ya que no hay mejoras visibles, y sí ocupa un tiempo que hace más lenta la ejecución del algoritmo.

	No. de reinciali-zaciones	Tiempo de reinicialización	Iteraciones	Tiempo total
Figura 6.16	0	0	13	93.7 s.
Figura 6.17	6	177.8 s.	13	260 s.

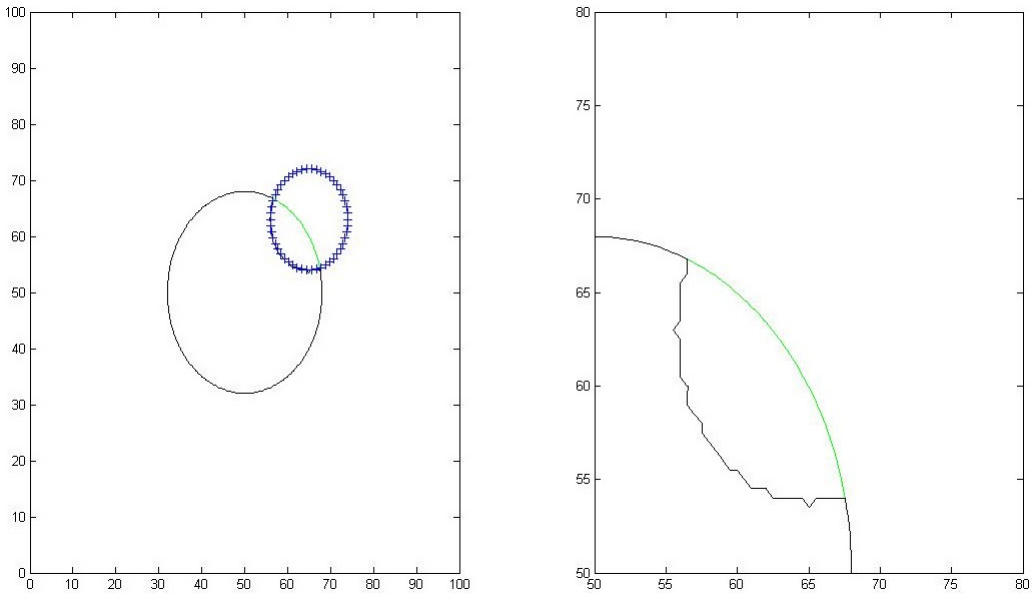


Figura 6.16: $\Delta t = 0.1$ y sin reinicialización.

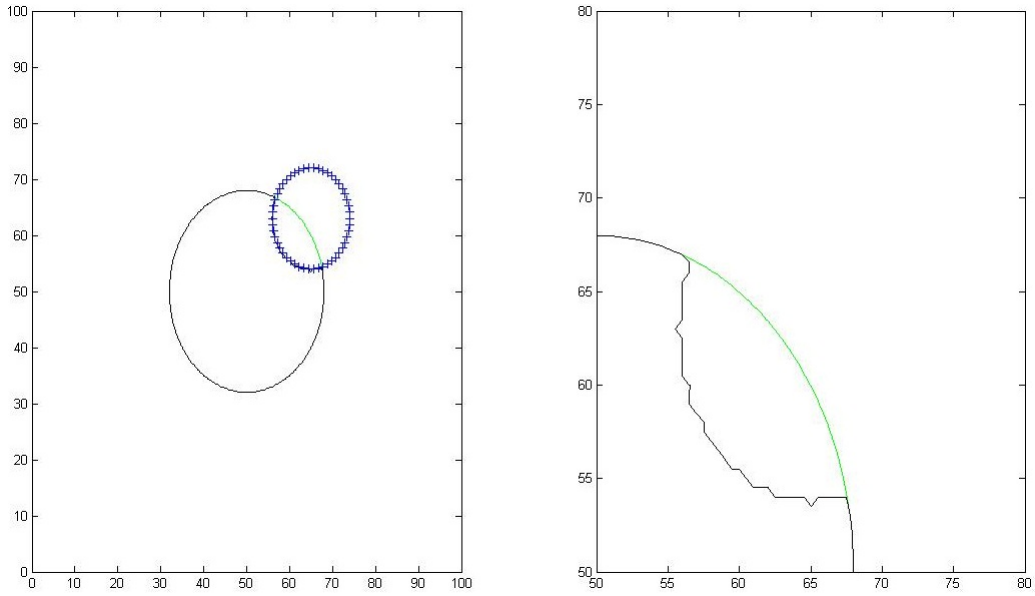


Figura 6.17: $\Delta t = 0.1$, con reinicialización en cada iteración.

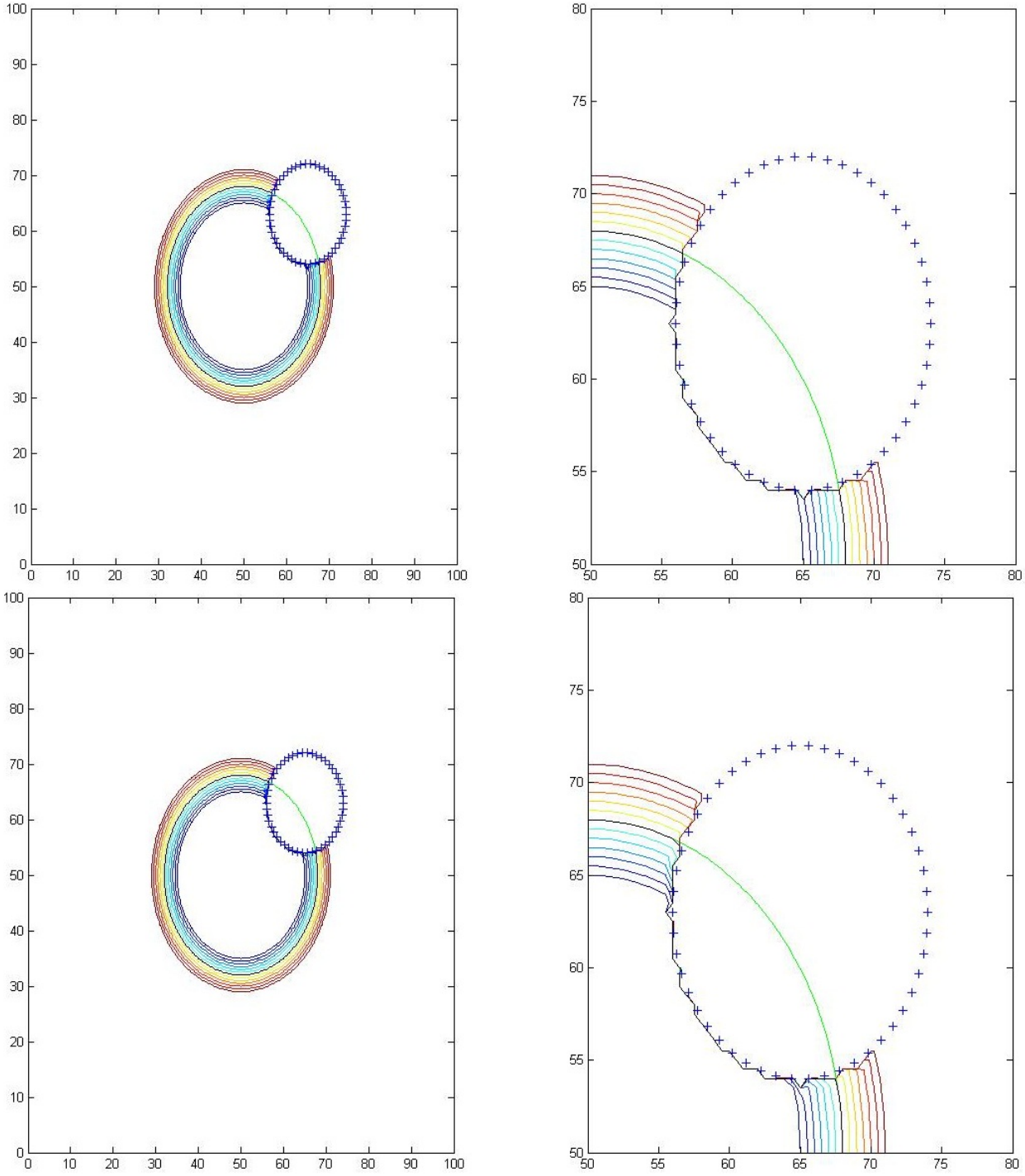


Figura 6.18: (*arriba*) sin reinicialización.(*abajo*) con reinicialización cada 2 iteraciones.

	No. de reinicializaciones	Tiempo de reinicialización	Iteraciones	Tiempo total
Figura 6.19	0	0	97	388.75 s.
Figura 6.20	9	260 s.	97	631 s.

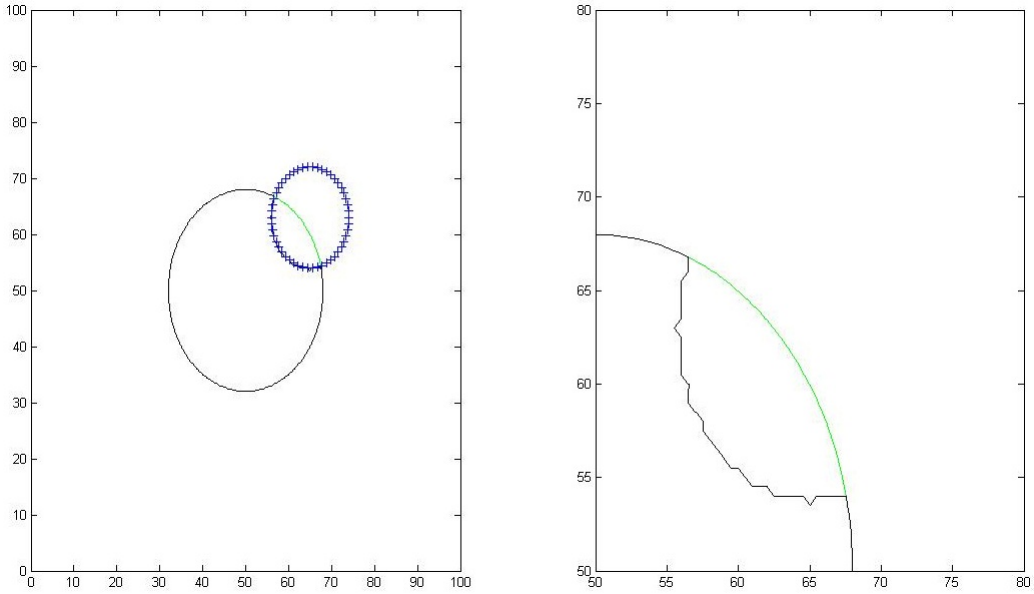


Figura 6.19: $\Delta t = 0.01$ y sin reinicialización.

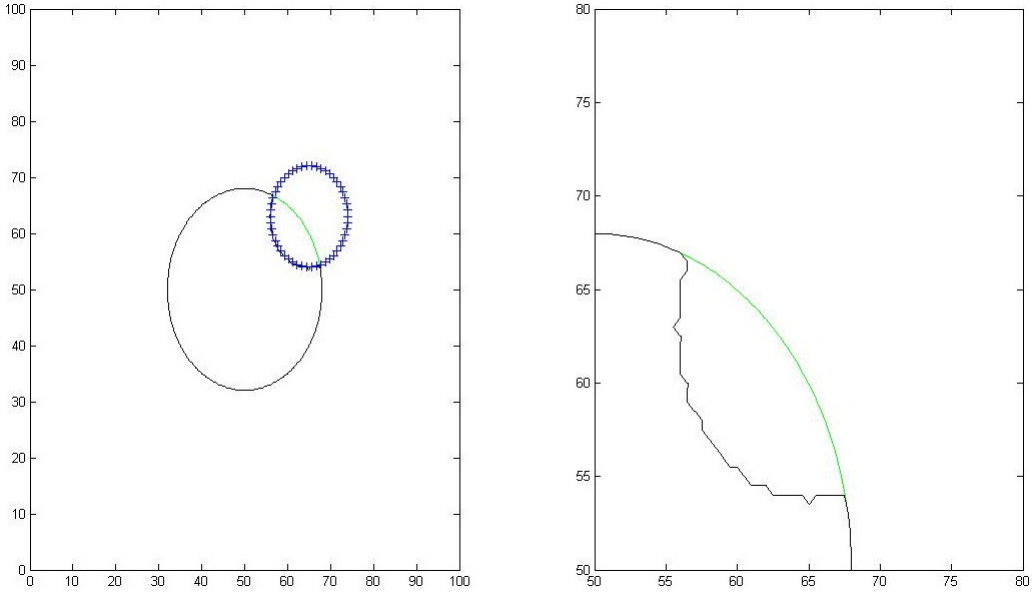


Figura 6.20: $\Delta t = 0.01$, con reinicialización en cada iteración.

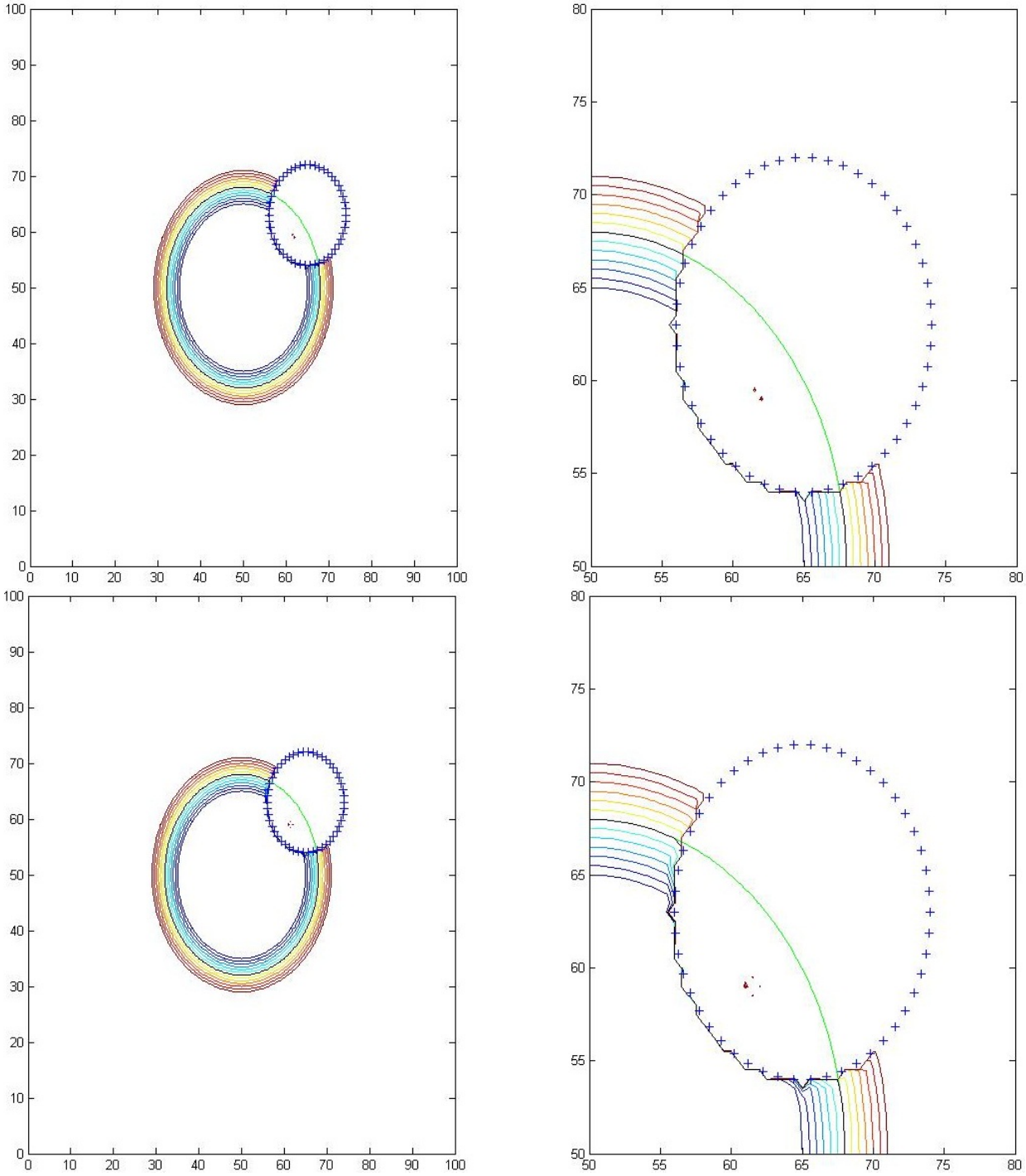


Figura 6.21: (*arriba*) sin reinicialización.(*abajo*) con reinicialización cada 10 iteraciones.

A pesar de que se trabaja con valores de Δt pequeños, las muescas o pequeñas puntas que quedan después de la deformación local no desaparecen, e incluso los conjuntos de nivel dentro de la herramienta durante el proceso empeoran un poco dejando las marcas que se notan en la Figura 6.21.

Ahora se trabaja con una herramienta de forma cuadrada, dejando igual todo lo demás. Se hacen

primeramente las pruebas con $\Delta t = 1$. La punta de la herramienta y las partes planas de la misma, se dejan plasmadas al deformarse el objeto. En las dos situaciones mostradas en las figuras 6.22 y 6.23, se tienen similares resultados. La reinicialización mejora ligeramente la impresión que deja el cuadrado haciéndo los bordes un poco más uniforme. Nuevamente, los conjuntos de nivel que se consideran se apilan muy estrechamente cerca de la interface, luego de la deformación. Así, no se saca provecho realmente de la reinicialización, y nuevamente se utiliza un gran tiempo en ello.

	No. de reinciali-zaciones	Tiempo de reinicialización	Iteraciones	Tiempo total
Figura 6.22	0	0	3	133 s.
Figura 6.23	3	91.8 s.	3	335.05 s.

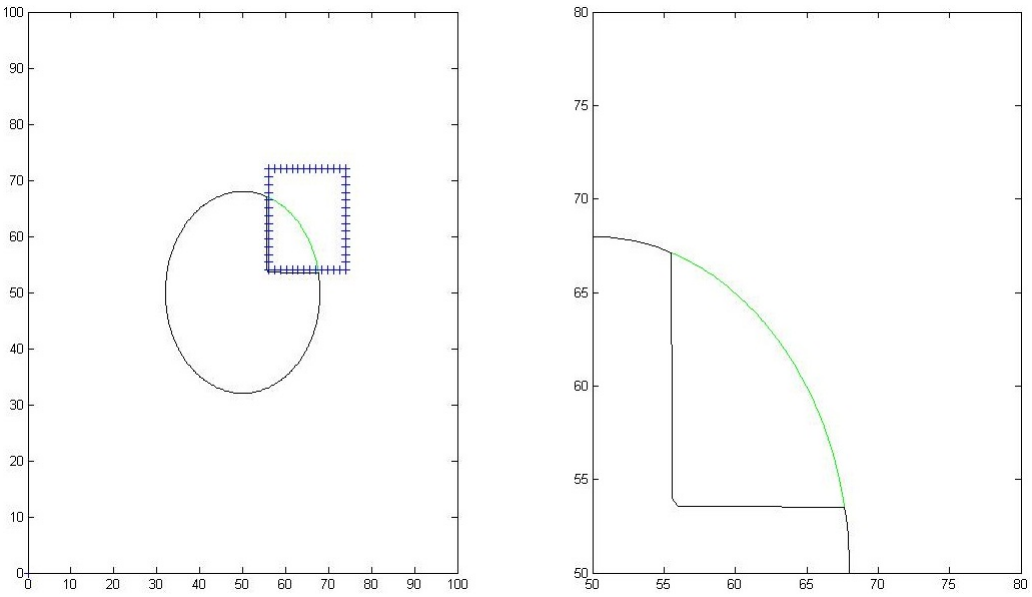


Figura 6.22: $\Delta t = 1$ y sin reinicialización.

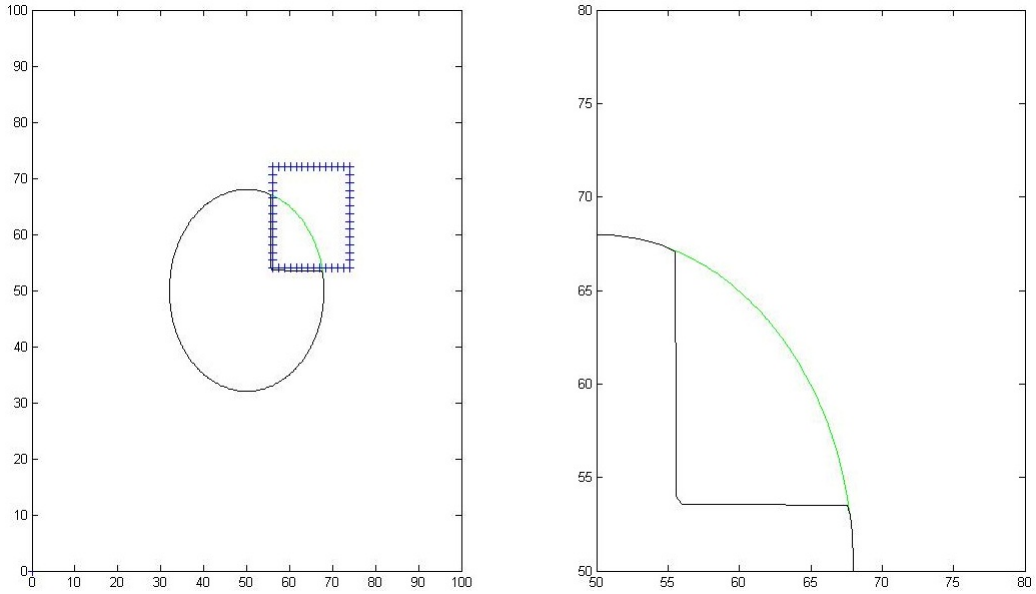


Figura 6.23: $\Delta t = 1$, con reinicialización en cada iteración.

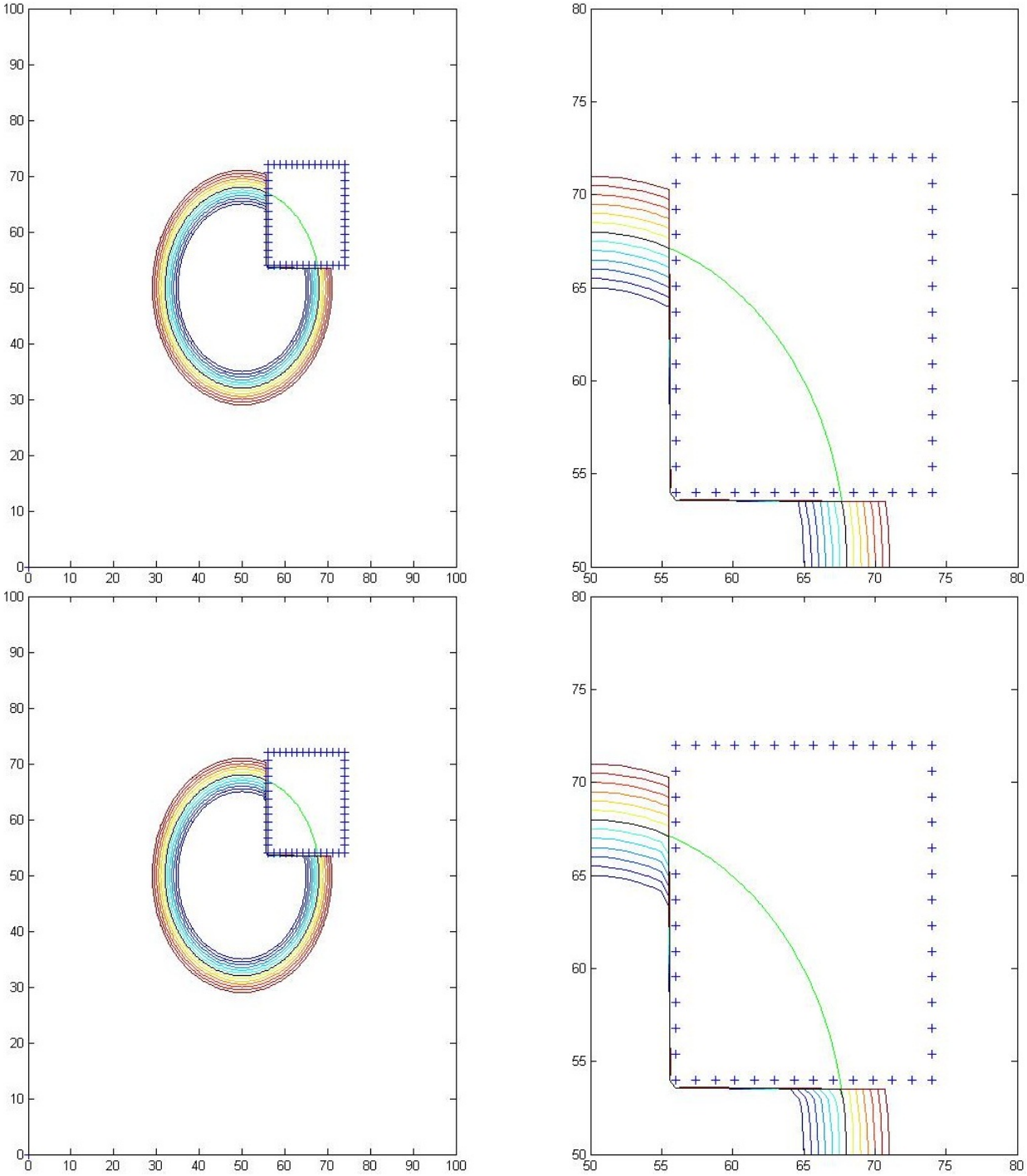


Figura 6.24: (*arriba*) sin reinicialización.(*abajo*) con reinicialización en cada iteración.

Ahora, en el caso en que $\Delta t = 0.1$ y 0.01, tanto con reinicialización como sin ella, la forma de la interface no queda como se esperaba, además que al final se tiene una parte no conexa en el lugar en que debía de quedar la forma de la herramienta. No se tiene una respuesta del por qué sucede esto, pero parece ser que los conjuntos de nivel se pegan más rápido a los bordes de la herramienta, de lo que la

parte central avanza, dejando al final una parte aislada.

La razón por la que no desapareció la parte disconexa de la interface, es por la forma en que se deja de aplicar la deformación, ya que con algunas iteraciones más desaparecería. Hay que recordar que el algoritmo, mientras exista algún punto de la superficie de la herramienta que tenga un valor no positivo, las ecuaciones de movimiento normal se siguen aplicando hasta que todos tengan valores positivos, esto deja fuera los puntos dentro de la interface, que no son tomados en cuenta sus valores. Extendiendo la verificación de los valores de *phi* en la parte interior de la herramienta, hubiera hecho que esta fuera eliminada.

	No. de reinciali-zaciones	Tiempo de reinicialización	Iteraciones	Tiempo total
Figura 6.25	0	0	7	174 s.
Figura 6.26	4	116.7 s.	8	218.8 s.

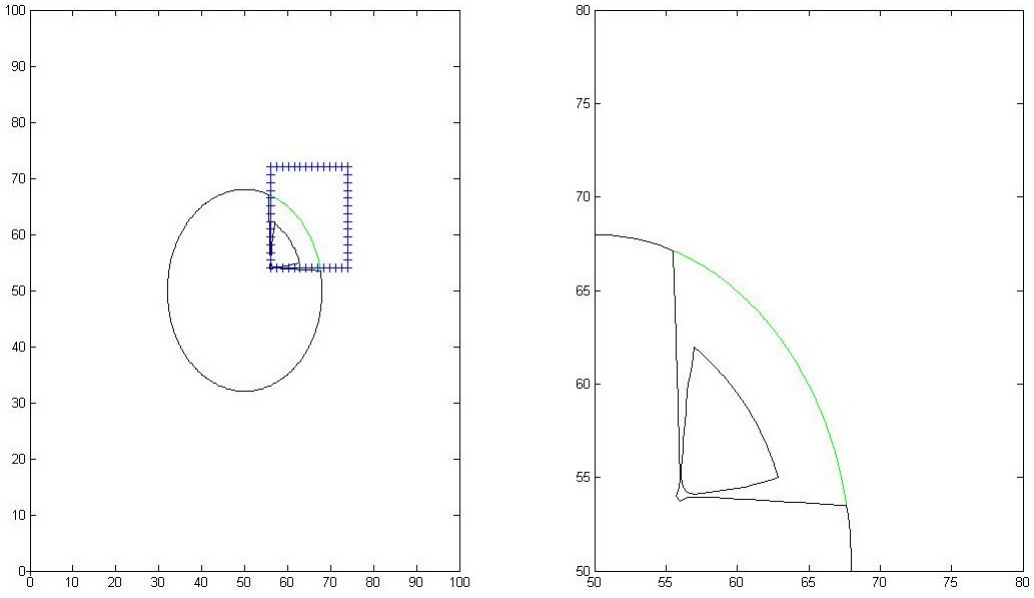


Figura 6.25: $\Delta t = 0.1$ y sin reinicialización.

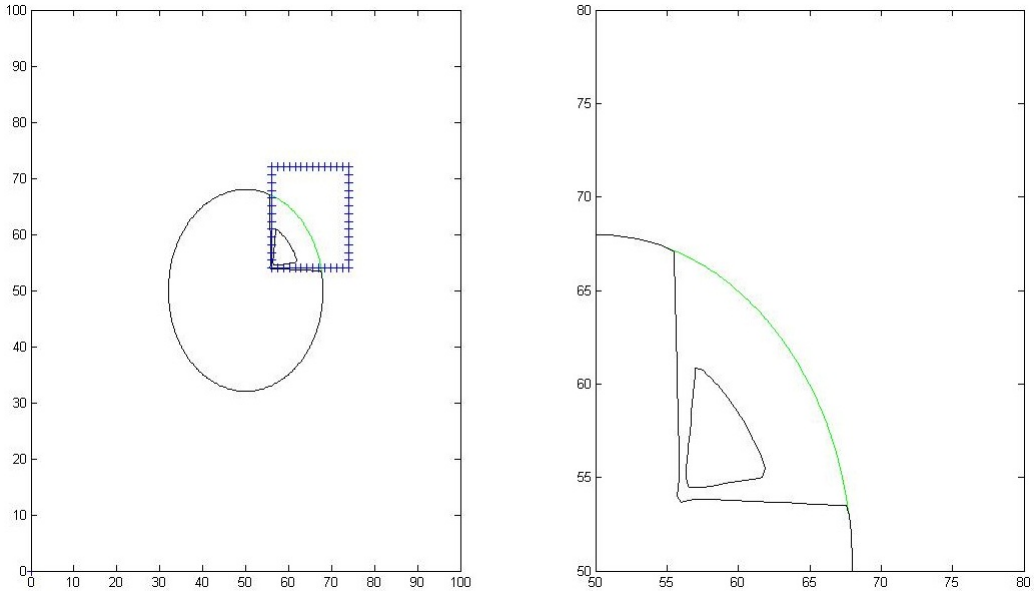


Figura 6.26: $\Delta t = 0.1$, con reinicialización en cada 2 iteraciones.

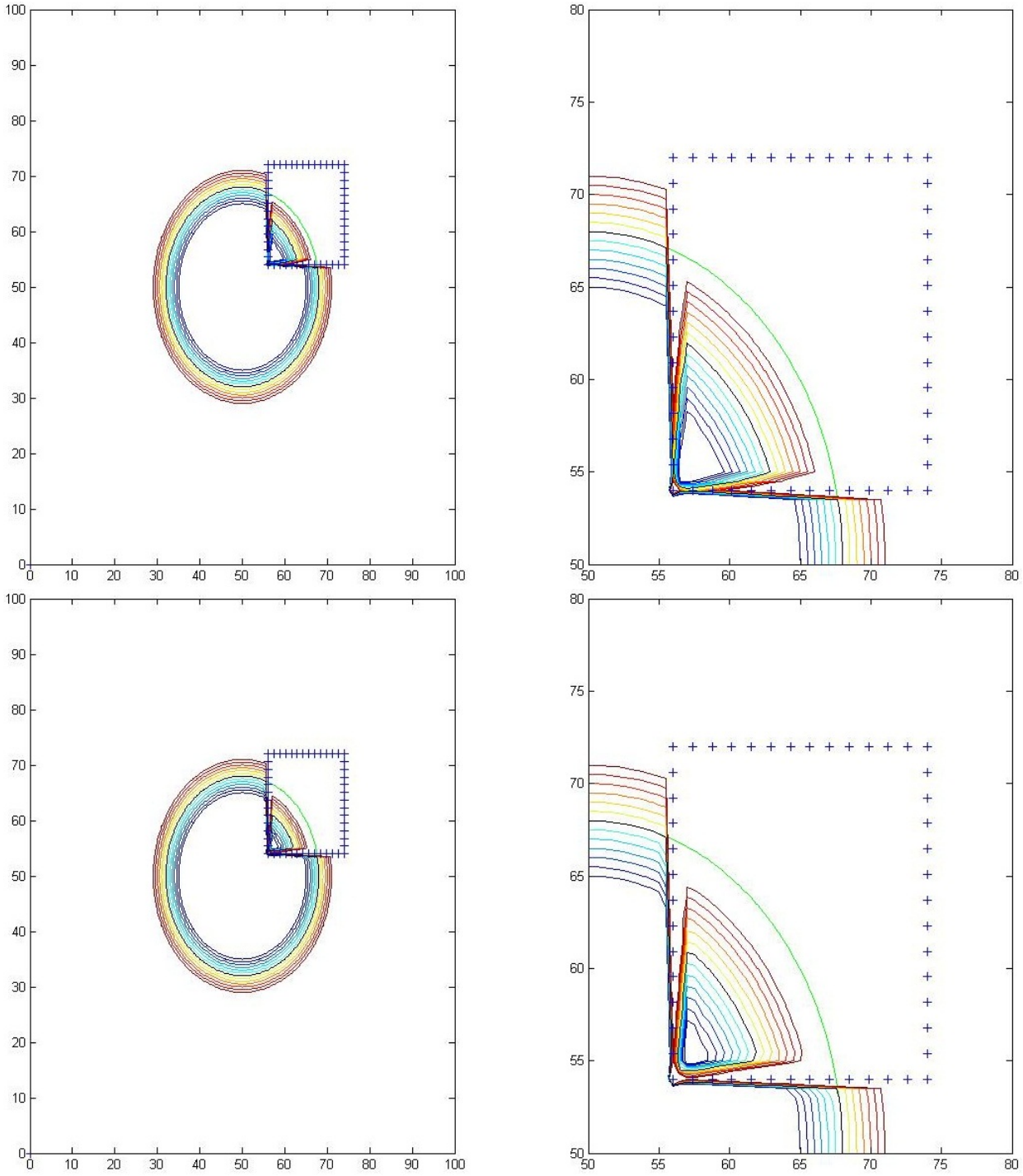


Figura 6.27: (*arriba*) sin reinicialización.(*abajo*) con reinicialización en cada 2 iteraciones.

	No. de reinicializaciones	Tiempo de reinicialización	Iteraciones	Tiempo total
Figura 6.28	0	0	55	510 s.
Figura 6.29	14	410.7 s.	71	862 s.

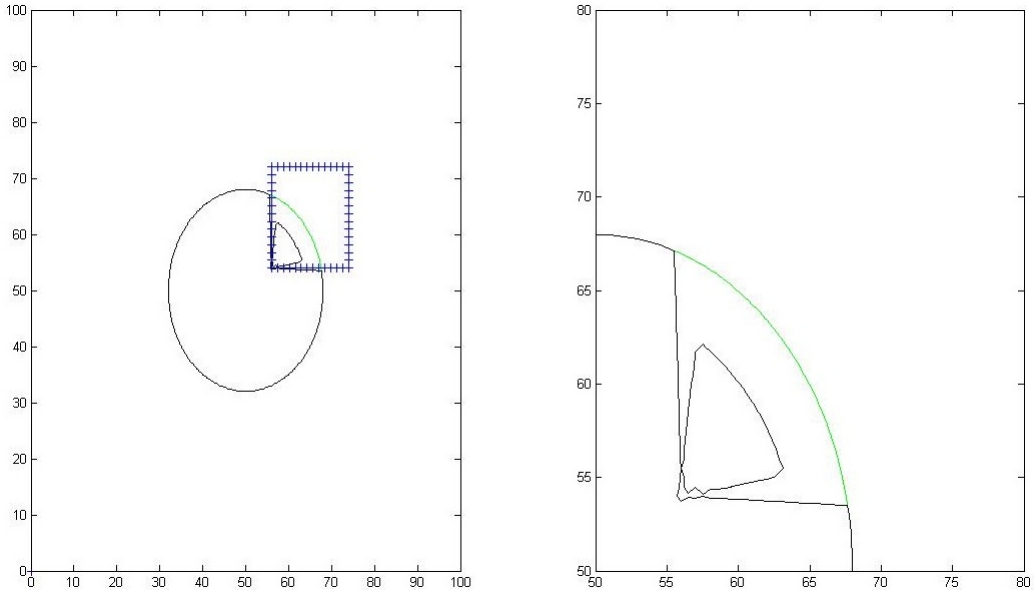


Figura 6.28: $\Delta t = 0.01$ y sin reinicialización.

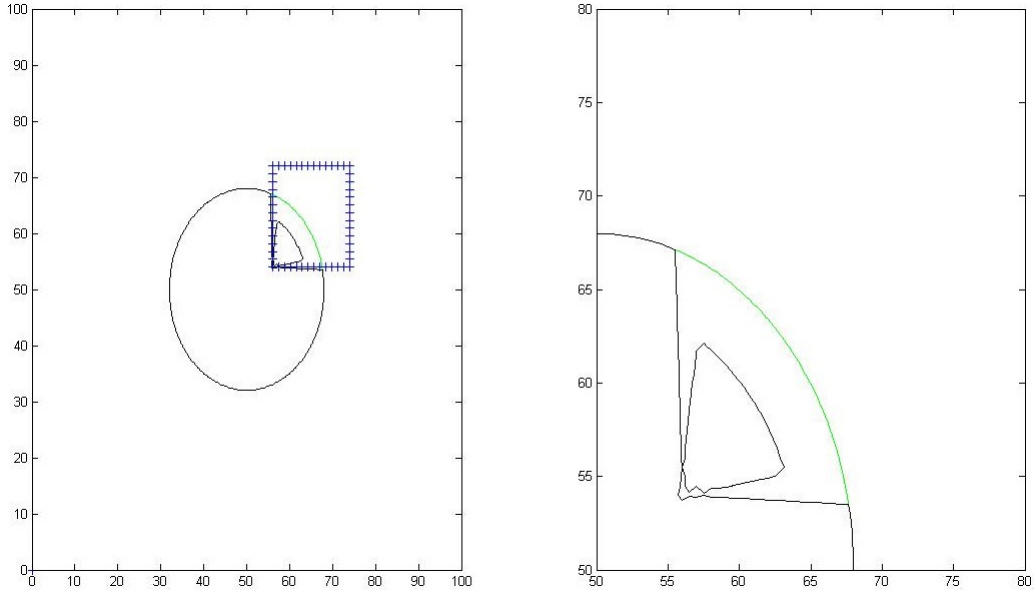


Figura 6.29: $\Delta t = 0.01$, con reinicialización en cada 5 iteración.

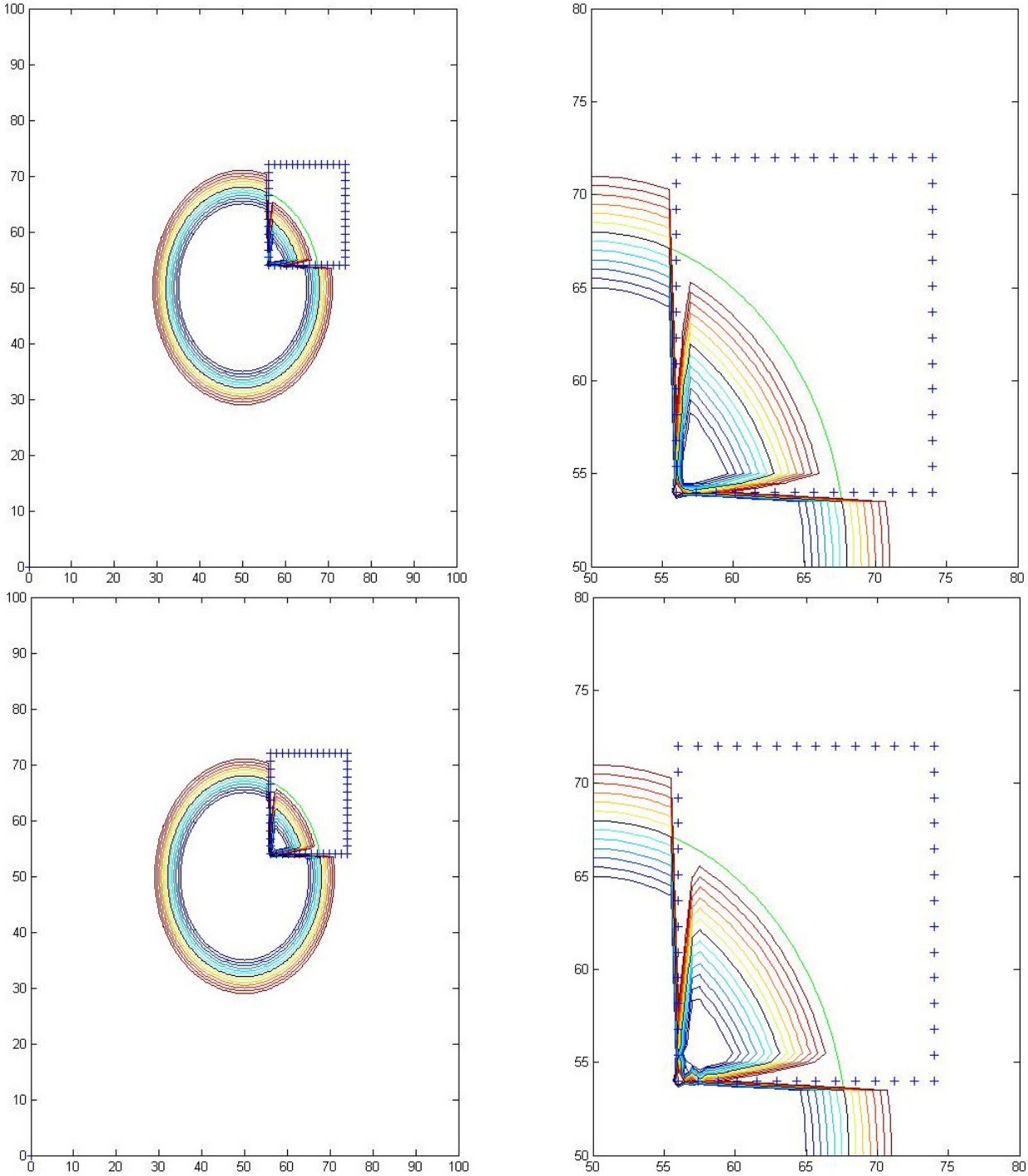


Figura 6.30: (*arriba*) sin reinicialización.(*abajo*) con reinicialización en cada 5 iteraciones.

Ahora se muestran ejemplos de deformación local para un cuadrado. Se fija $\Delta t = 1$y las especificaciones de las herramientas utilizadas son las mismas que anteriormente se dieron. Se exhiben los mismos detalles expuestos para los casos anteriores, como que se dejan partes aisladas de los conjuntos de nivel y de la misma interface, aunque el ejemplo con la herramienta circular se comporta bien y cumple completamente

con el objetivo de dejar la forma de su contorno que tuvo contacto con el material.

	No. de reinciali-zaciones	Tiempo de reinicialización	Iteraciones	Tiempo total
Figura 6.31	0	0	3	133 s.
Figura 6.32	3	91.8 s.	3	335.05 s.

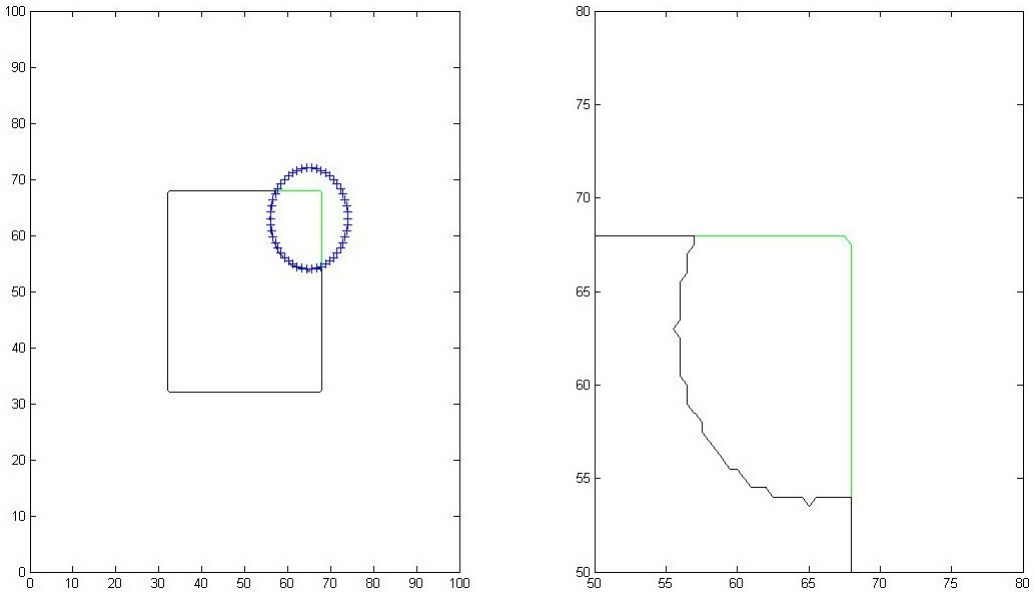


Figura 6.31: $\Delta t = 1$ y sin reinicialización.

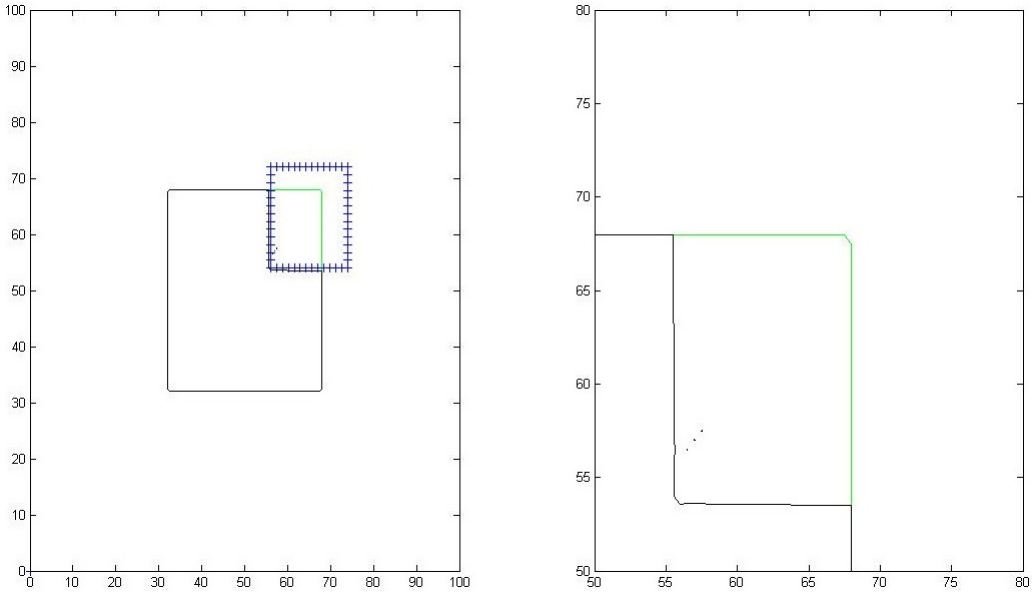
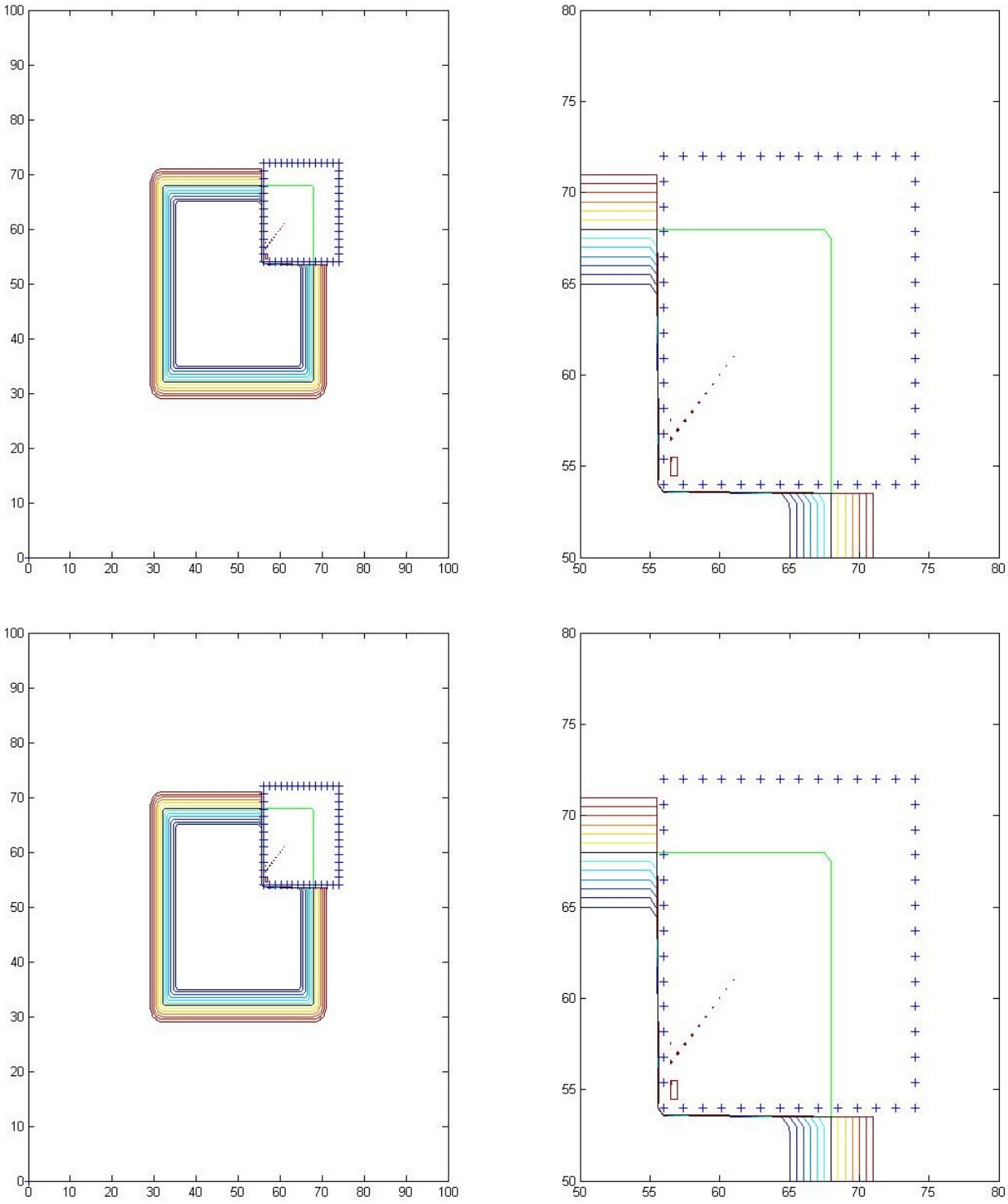


Figura 6.32: $\Delta t = 1$ y sin reinicialización.



El proceso de deformación local cuenta con muchos detalles, que posiblemente puedan ser corregidos con otras aplicaciones o con versiones mejoradas de métodos expuestos en este trabajo. Además si el objetivo es dejar la marca de la herramienta, también podría hacerse de otra manera. Por ejemplo, si se tiene una representación explícita de la herramienta, podría hallarse una representación imlícita aplicando un método específico, luego se procedería a realiar una operación booleana de resta y la herramienta

dejaría su forma en el objeto deformable.

6.3. Compensación de volumen

La compensación de volumen es muy parecida a la deformacion local, en este caso la zona que interesa ya no es la ocupada por la herramienta, sino la que ocupa su complemento. Se forma la función indicadora del exterior de la herramienta, χ_{Γ^c}, al igual que se hizo con la función indicadora de la misma, sólo asignando el valor contrario de los únicos dos que pueden tomar estas funciones. La otra diferencia que se tiene, es que en este caso la ecuación de deformación tiene el factor $(Vol_{\Gamma^c}(\phi) - c)\delta(\phi)$ o $(Vol_{\Gamma^c}(\phi) - c)||\nabla\phi||$, según sea el caso.

Para realizar los experimentos se tomó como herramienta una circunferencia de radio 9, con centro en (65,63), y como objeto deformable otra circunferencia de radio 18 y centro (50,50). Se fija el paso espacial en $h = 0.5$ con 201 puntos discretos en cada eje coordenado, y se varía el valor de paso de tiempo $\Delta t = 1, 0.1$ y 0.01, utilizando tanto la función delta como el gradiente, también se verifica que pasa cuando se le incorpora la reinicialización.

Los primeros resultados que se obtuvieron fue utilizando $Deltat = 1$ y $\delta(\phi)$ para la deformación, comparando de este modo cómo afecta la reinicialización. El primer par de figuras muestran a la herramienta definida visualmente con cruces en su contorno y el objeto que se deforma con una línea continua (en verde se muestra la interface inicial y la la línea negra la interface final). El segundo par de figuras, además de la interface de nivel cero, muestra, en una banda de líneas de colores, los conjuntos de nivel con valores de -3 a 3, espaciados de 0.5 en 0.5.

Las tablas que acompañan a las figuras muestran algunos datos importantes como el número de iteraciones de actualización en la evolución de la interface, el número de reinicializaciones, el tiempo de reinicialización y el tiempo de ejecución total en segundos. La última columna correspondiente a $|Vol_{\Gamma^c}(\phi) - c|$, indica la diferencia entre el volumen inicial y el volumen de la última actualización de la evolución, lo que se espera sea pequeños. Esto último se considera debido a que se llega un momento en que tal diferencia ya no desciende más, se estanca u oscila entre positivo o negativo, esto quiere decir, que en ocasiones hay un exceso de volumen con respecto al original, por lo que la interface en estas sircunstancias se retrae. Lo ideal es que esos movimientos no sean grandes.

Como puede verse en las Figuras 6.33 y 6.34, la reinicialización hizo que la interface resultante (de color negro) sea un poco más uniforme, sin embargo se paga un precio caro en cuestión de tiempo, ya que sin reinicialización se lleva aproximadamente 71 segundos con un total de 12 iteraciones de evolución de la interface, mientras tanto con reinicialización se llevó a cabo en 483.5 segundos con 21 iteraciones de evolución, y un tiempo de reinicialización de 295.8 segundos, lo cual es más de la mitad del total del tiempo. En este caso no hay mucha diferencia entre uno y otro resultado, por lo que el costo de reincicialización no vale la pena. Lo que puede observarse de la Figura 6.35, es que la función δ repliega

los conjuntos de nivel en sus límites, dejándolos muy juntos, lo que podría ocasionar problemas en alguna futura evolución de la interface.

La ventaja que se tiene de la δ en este ejemplo, es que los valores en sus límites evita que la evolución de la interface sea inestable, a pesar de que el paso de tiempo dt y $|Vol_{\Gamma^c}(\phi) - c|$ sean grandes. Por lo contrario, cuando se utiliza la norma del gradiente $\nabla\phi$ en la evolución, es decir, se emplea la ecuación (5.8), el movimiento se hace tan inestable que no tiene caso visualizarlo.

	No. de reinciali-zaciones	Tiempo de reinicialización	Iteraciones	Tiempo total	$\lvert Vol_{\Gamma^c}(\phi) - c\rvert$
Figura 6.33	0	0	12	71.7 s.	2
Figura 6.34	10	295.8 s.	21	483.5 s.	0.5

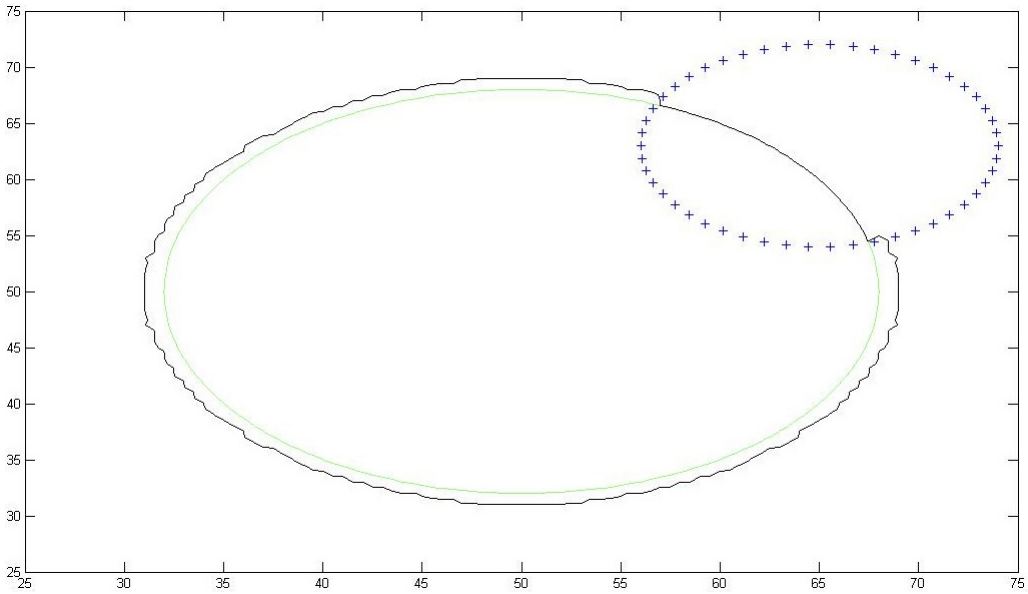


Figura 6.33: $\Delta t = 1$, con $\delta(\phi)$, sin reinicialización.

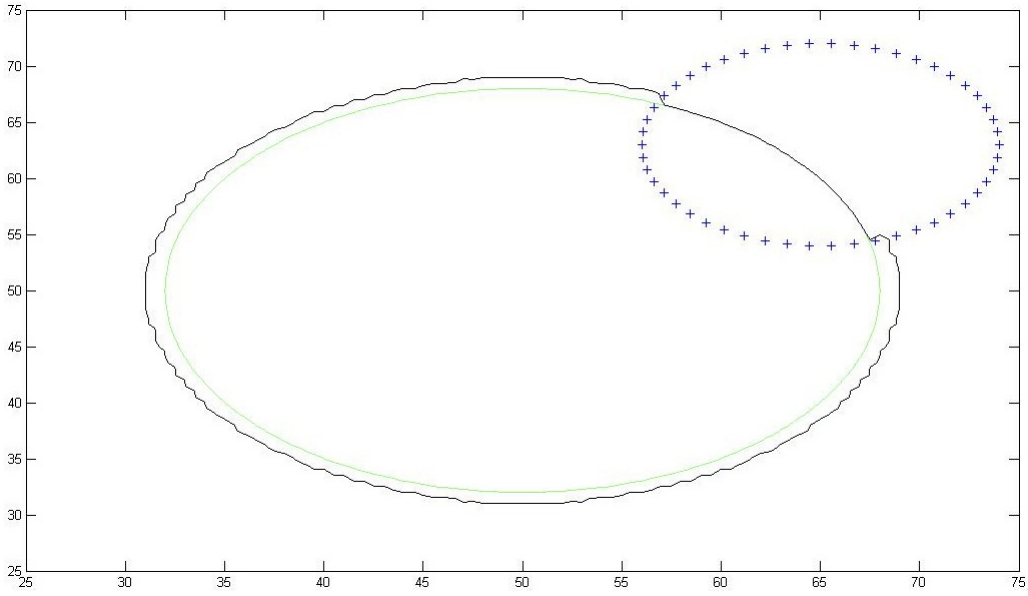


Figura 6.34: $\Delta t = 1$, con $\delta(\phi)$ y reinicialización cada 2 iteraciones.

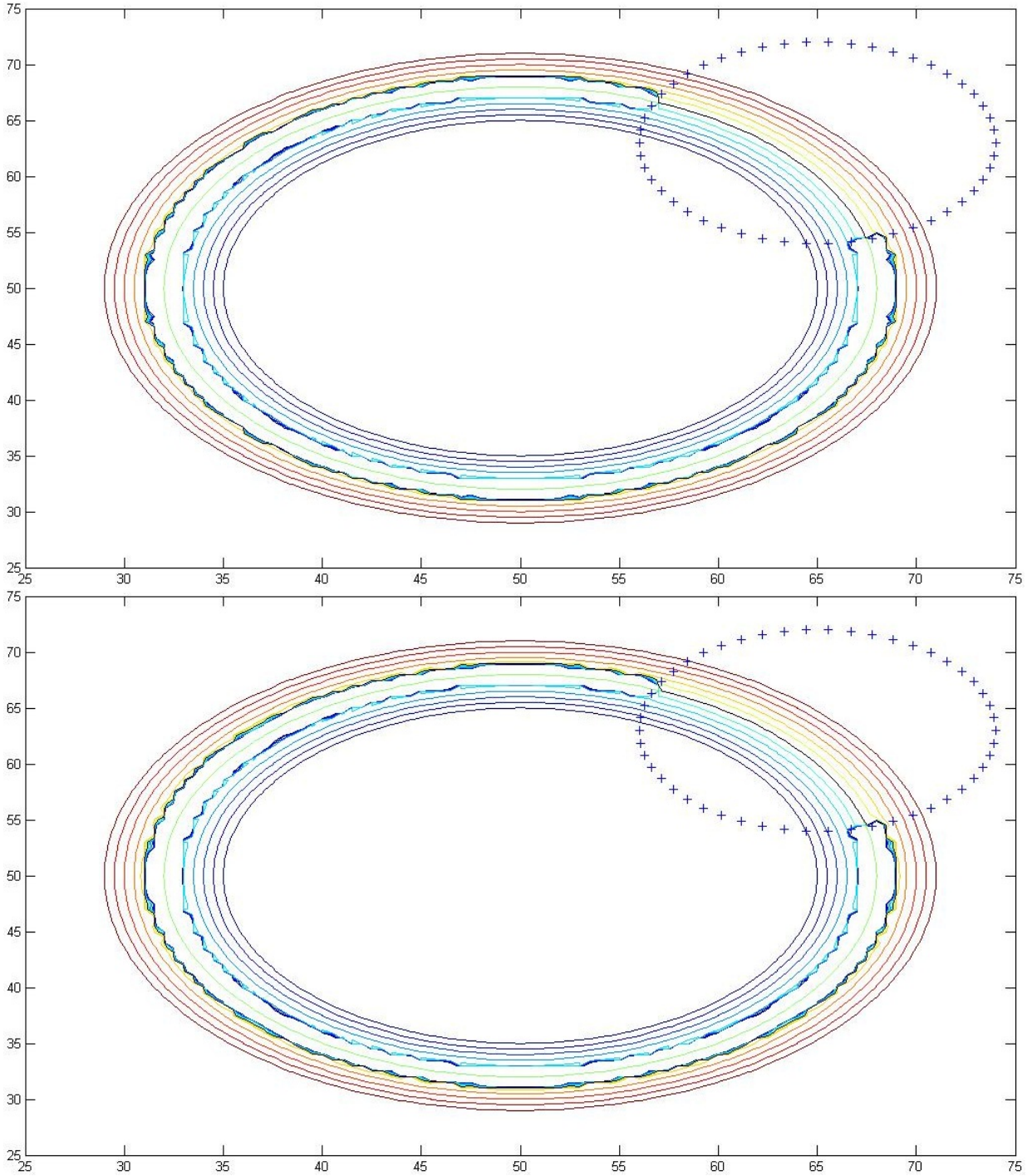


Figura 6.35: (*arriba*) sin reinicialización.(*abajo*) con reinicialización cada 2 iteraciones.

Ahora si modificamos el paso de tiempo Δt reduciéndolo a 0.1, ¿será estable la discretización de la ecuación (5.8)? Desafortunadamente no, todavía sigue siendo inestable, el menor valor de Δt no es suficiente para atenuar el valor de la diferencias $|Vol_{\Gamma^c}(\phi) - c|$. Sin embargo, el movimiento de con la delta δ, parece mejorar incluso.

	No. de reinicializaciones	Tiempo de reinicialización	Iteraciones	Tiempo total	$\lvert Vol_{\Gamma^c}(\phi) - c\rvert$
Figura 6.36	0	0	22	138 s.	0.5
Figura 6.37	9	262.36 s.	18	363.65 s.	0.5

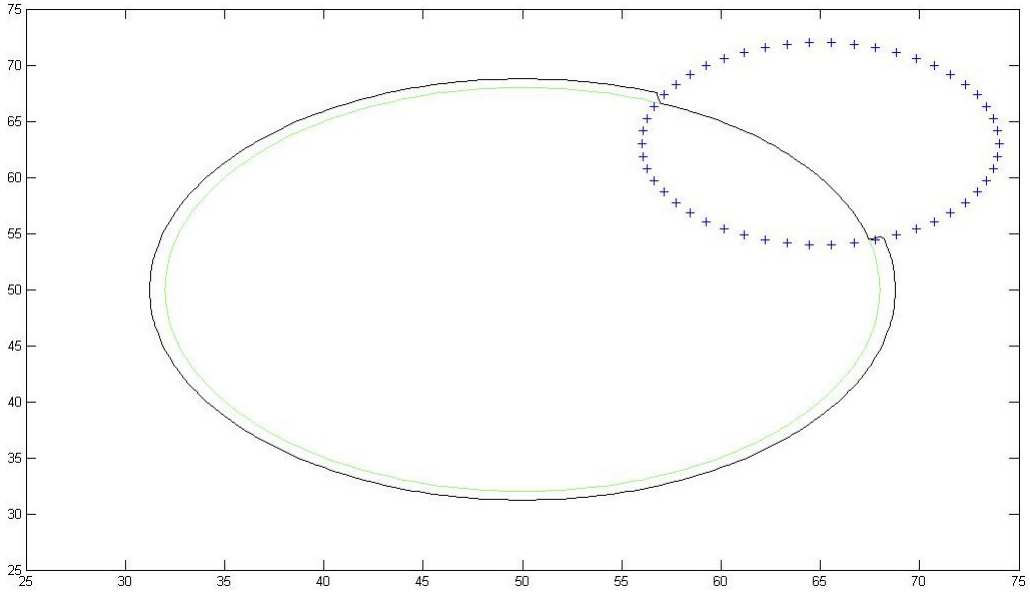


Figura 6.36: $\Delta t = 0.1$, con $\delta(\phi)$ sin reinicialización.

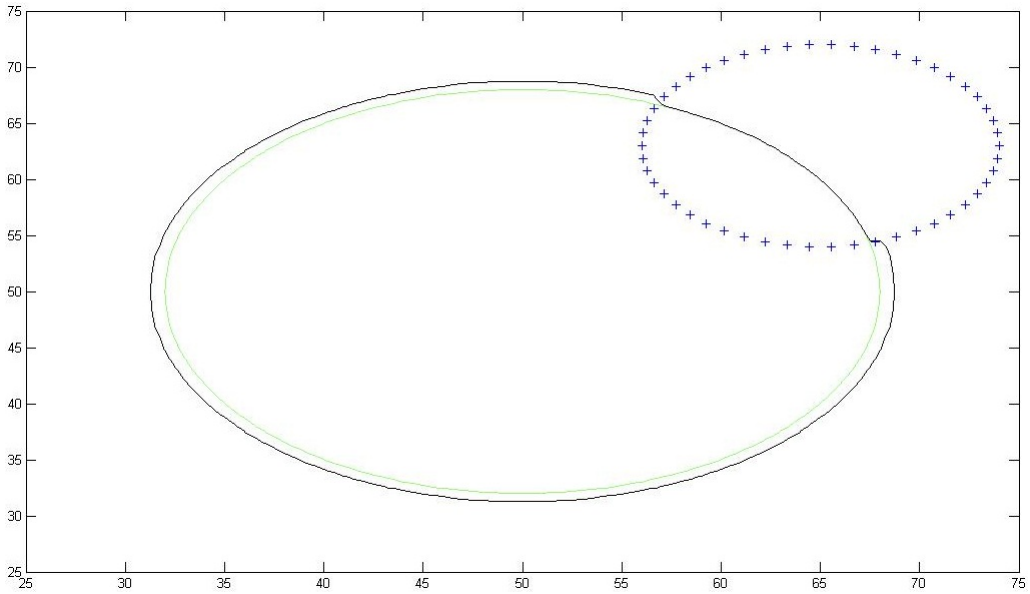


Figura 6.37: $\Delta t = 0.1$, con $\delta(\phi)$ con reinicialización cada 2 iteraciones.

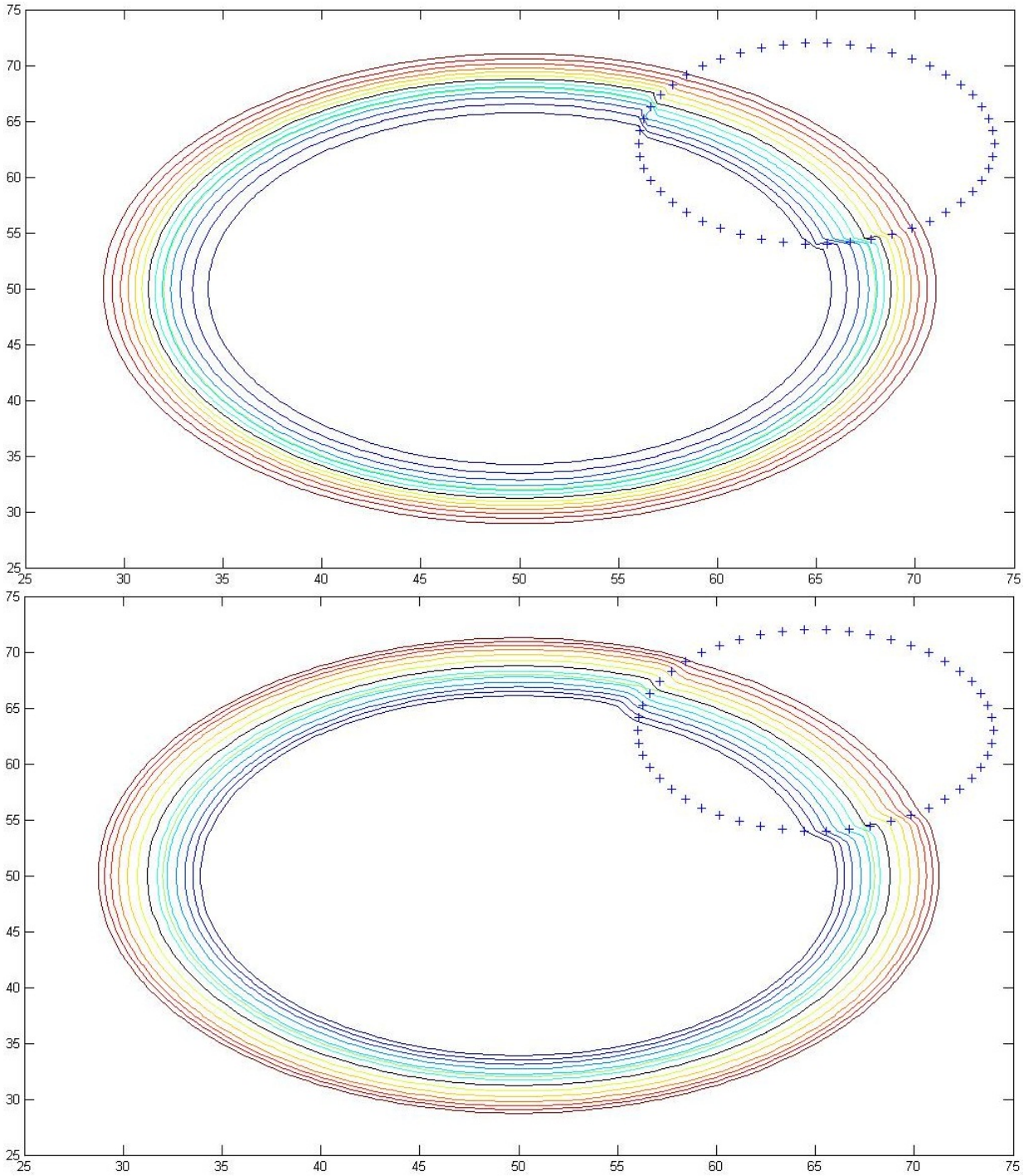


Figura 6.38: (*arriba*) sin reinicialización.(*abajo*) con reinicialización cada 2 iteraciones.

Casi no hay diferencias entre los resultados visuales, con y sin reinicialización. Donde sí parece haber mucha diferencia entre ambos, es en el tiempo total. A pesar que el tiempo total del caso sin reinicialización, con repecto al anterior valor de $\Delta t = 1$, resulta ser, con la reinicialización, más del doble del tiempo, y éste ocupa más de la mitad del tiempo de ejecución como se puede en la tabla correspondiente a este

caso ($\Delta t = 0.1$). Una diferencia visualmente palpable, pero tampoco de mucha importancia, además del costo de tiempo, es en la parte correspondiente a los conjuntos de nivel: al no reinicializarse, los conjuntos de nivel van separándose, lo que no ocurre al considerarse la reinicialización. Otro punto que hay que notar, es que al volverse menos brusco el movimiento debido al factor $\Delta t = 0.1$, el repliegamiento ya no se observa.

Ahora se presentan los casos cuando $\Delta t = 0.01$. Se sigue poniendo de manifiesto en las tablas que siguen, que el costo de la reinicialización sigue siendo alto, simplemente hay que reducir su utilización a lo menos posible, ya que por muy frecuente que se utilice, como es en todos los casos en esta sección, no mejora demasiado la evolución de la interface.

La función δ tampoco parece mejorar, si acaso, los conjuntos de nivel parecen ser más uniformes y espaciados, pero muy ligeramente. En cambio, el movimiento de la interface definitivamente se pudo hacer más estable esta vez con $||\phi||$ y (5.8), e incluso supera los resultados de la función δ, visualmente y con respecto al tiempo, tanto sin reinicialización como con ella. Es más, se llegó a una diferencia entre el volumen inicial y el actual de 0.1 que es mejor que 2, que como se ve en la tabla es lo que se obtiene al utilizar la δ.

	No. de reinciali-zaciones	Tiempo de reinicialización	Iteraciones	Tiempo total	$\lvert Vol_{\Gamma^c}(\phi) - c\rvert$
Figura 6.39	0	0	19	102 s.	2
Figura 6.40	9	234.95 s.	19	338 s.	2

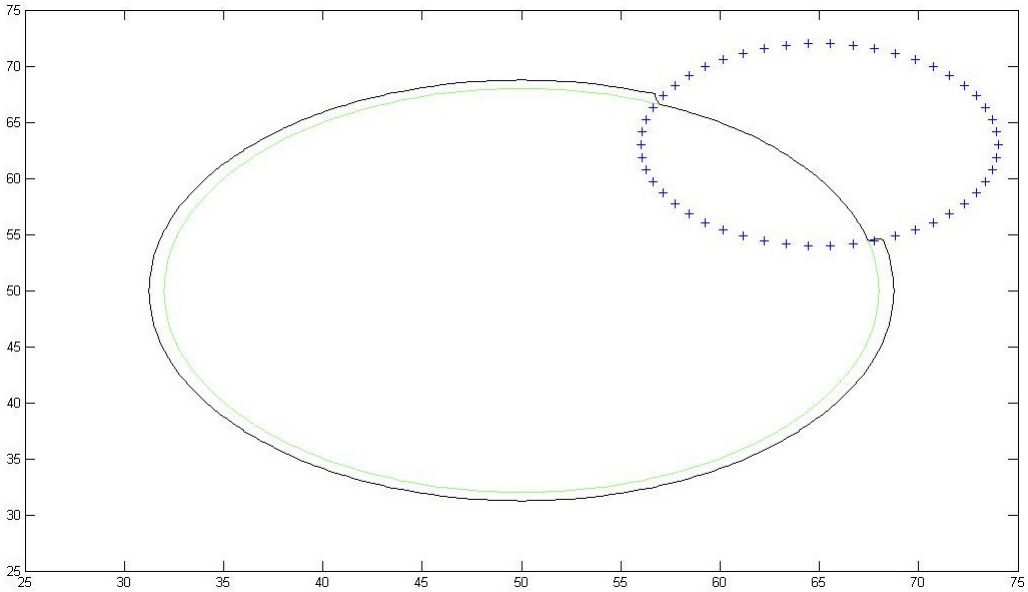


Figura 6.39: $\Delta t = 0.01$, con $\delta(\phi)$ sin reinicialización.

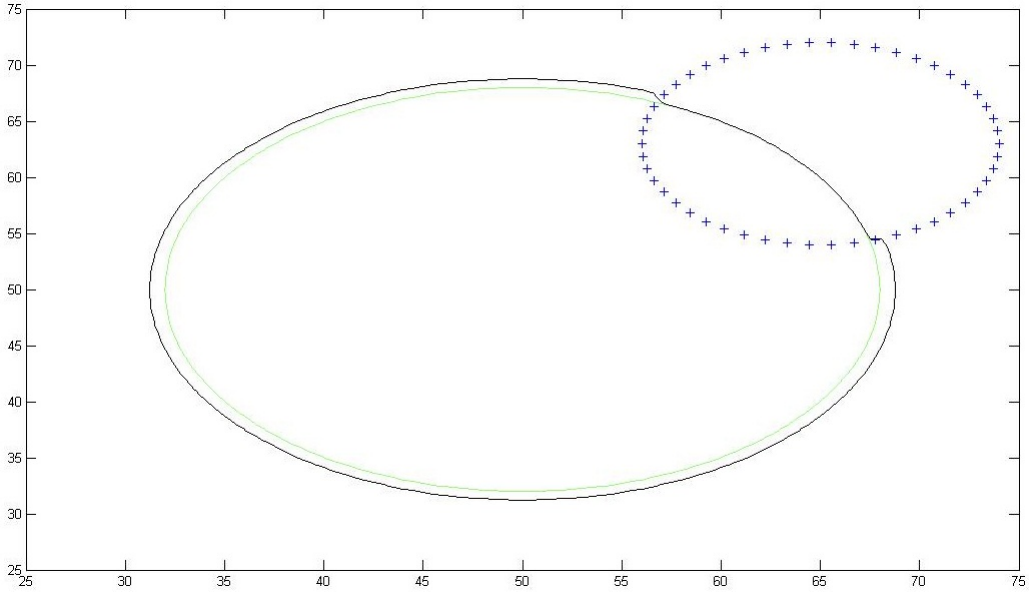


Figura 6.40: $\Delta t = 0.01$, con $\delta(\phi)$ con reinicialización cada 2 iteraciones.

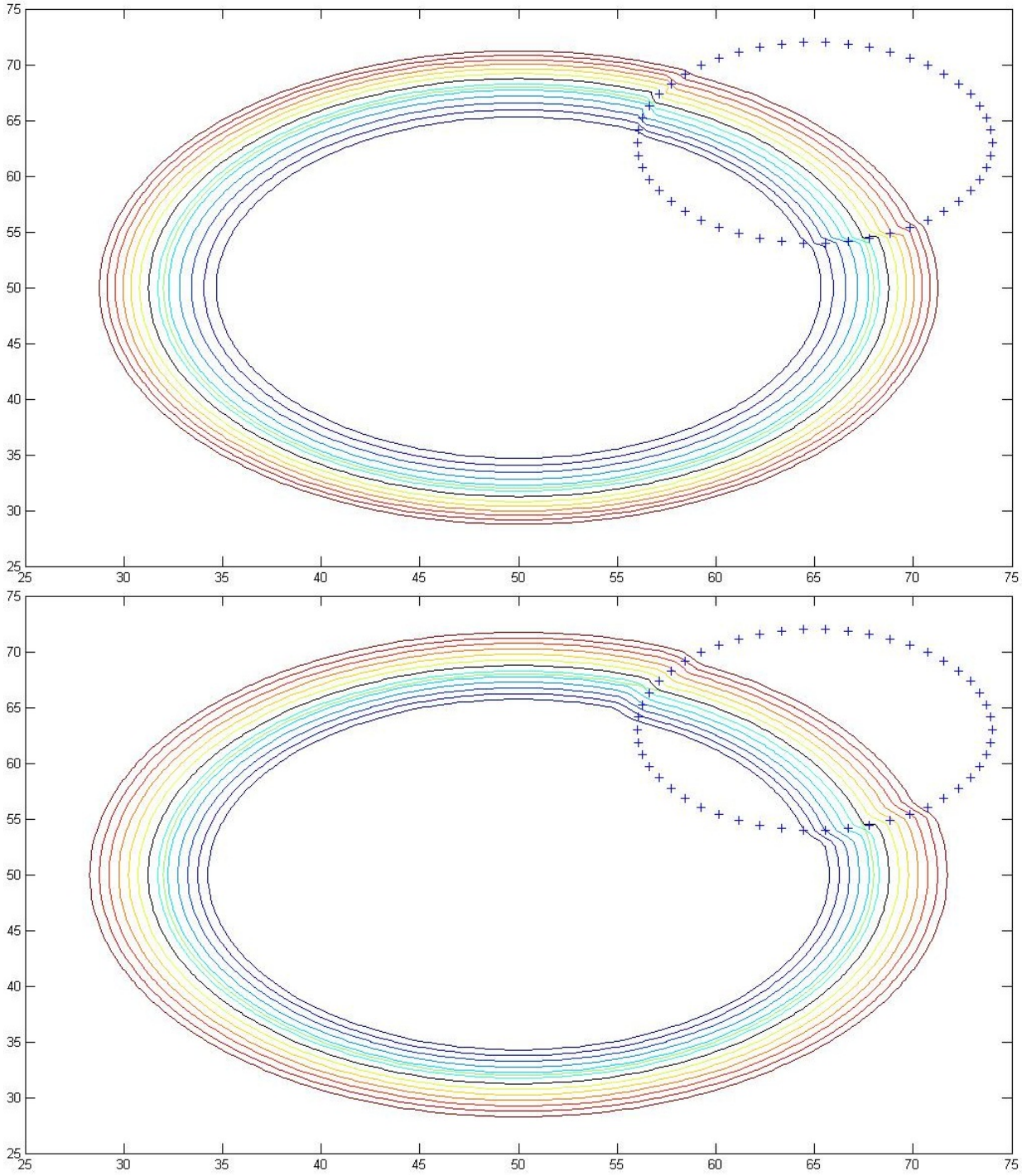


Figura 6.41: (*arriba*) sin reinicialización.(*abajo*) con reinicialización cada 2 iteraciones.

Si bien, los tiempos no empeoraron o siguieron igual con $||\nabla||$, lejos están de ser ideales, ya que como indica la siguiente tabla, la parte sin reinicialización, la que mejor tiempo tiene, es de casi dos minutos, y debe reducirse. En el futuro se puede estudiar otros métodos de convergencia para que el volumen tienda o se aproxime mucho mejor a cero. También se pueden estudiar métodos más convenientes o mejores que

los utilizadis en este trabao. Todavía hay mucho que sacar de este problema de compensación de volumen.

	No. de reinciali-zaciones	Tiempo de reinicialización	Iteraciones	Tiempo total	$\|Vol_{\Gamma^c}(\phi) - c\|$
Figura 6.42	0	0	17	102 s.	0.5
Figura 6.43	8	206.96 s.	17	290 s.	0.5

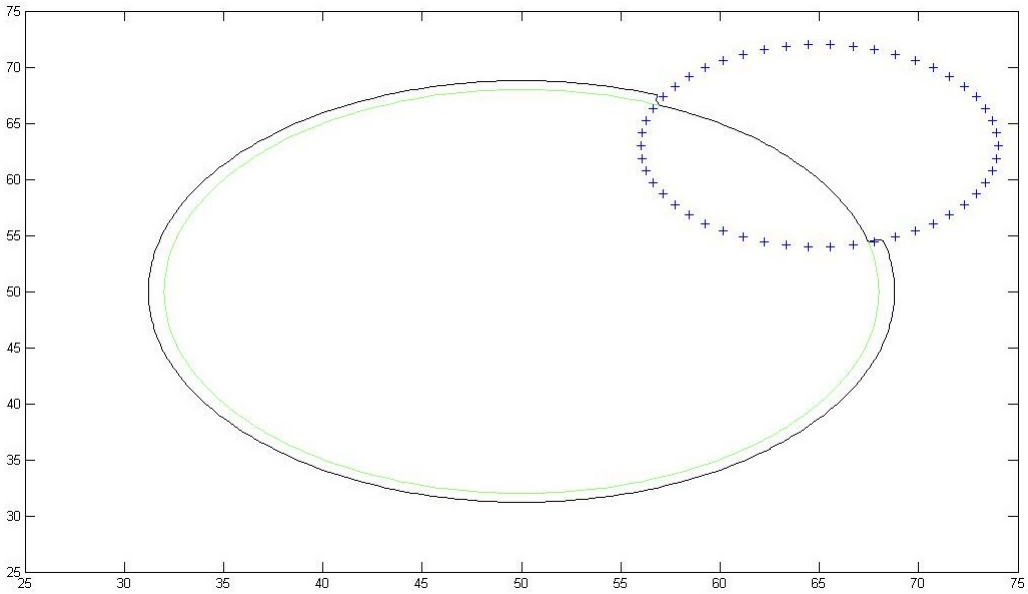


Figura 6.42: $\Delta t = 0.01$, con $||\nabla(\phi)||$ sin reinicialización.

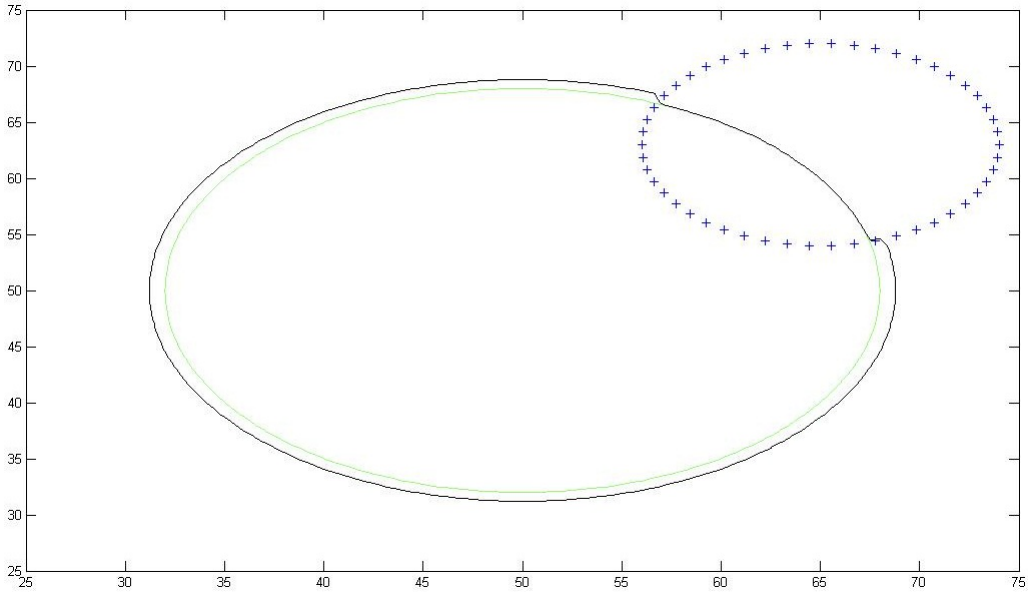


Figura 6.43: $\Delta t = 0.01$, con $||\nabla(\phi)||$ y reinicialización cada 2 iteraciones.

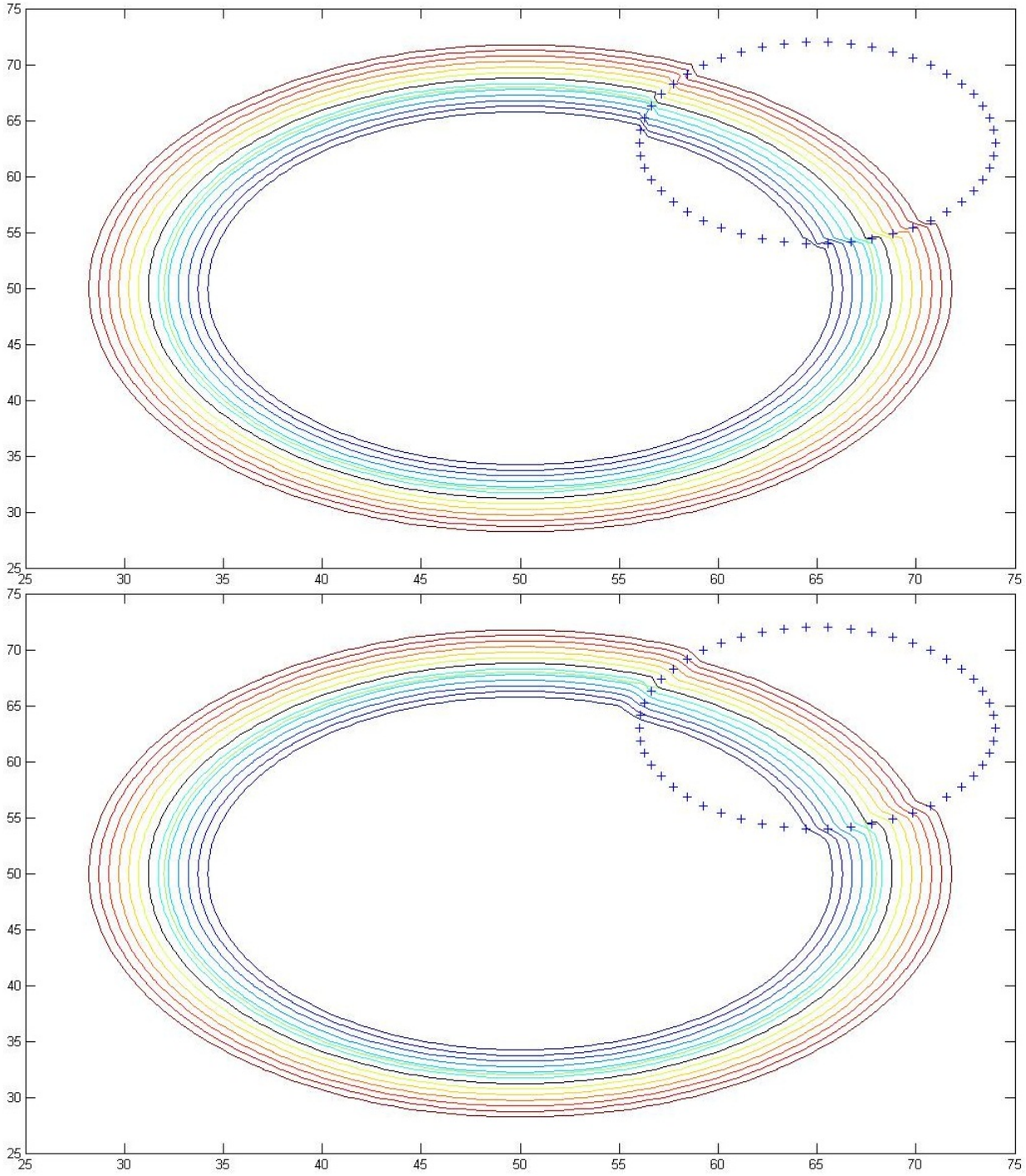


Figura 6.44: (*arriba*) sin reinicialización.(*abajo*) con reinicialización cada 2 iteraciones.

También se encontraron los resultados, para el caso $\Delta t = 0.01$ en otros tipos diferentes de combinaciones entre herramientas y objetos deformables. Cada objeto correspondiente está en la misma posición que el ejemplo utilizado para hallar los resultados anteriores.

	No. de reinciali-zaciones	Tiempo de reinicialización	Iteraciones	Tiempo total	$\lvert Vol_{\Gamma^c}(\phi) - c\rvert$
Figura 6.45	0	0	13	71.83 s.	0.75
Figura 6.46	6	154 s.	13	223.6 s.	0.75

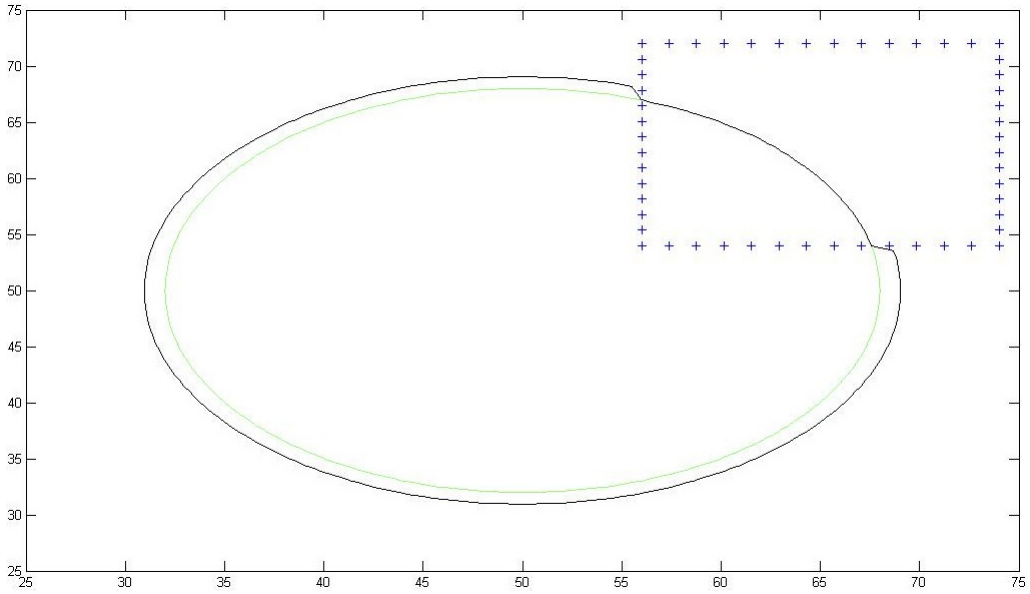


Figura 6.45: $\Delta t = 0.01$, con $||\nabla(\phi)||$ sin reinicialización.

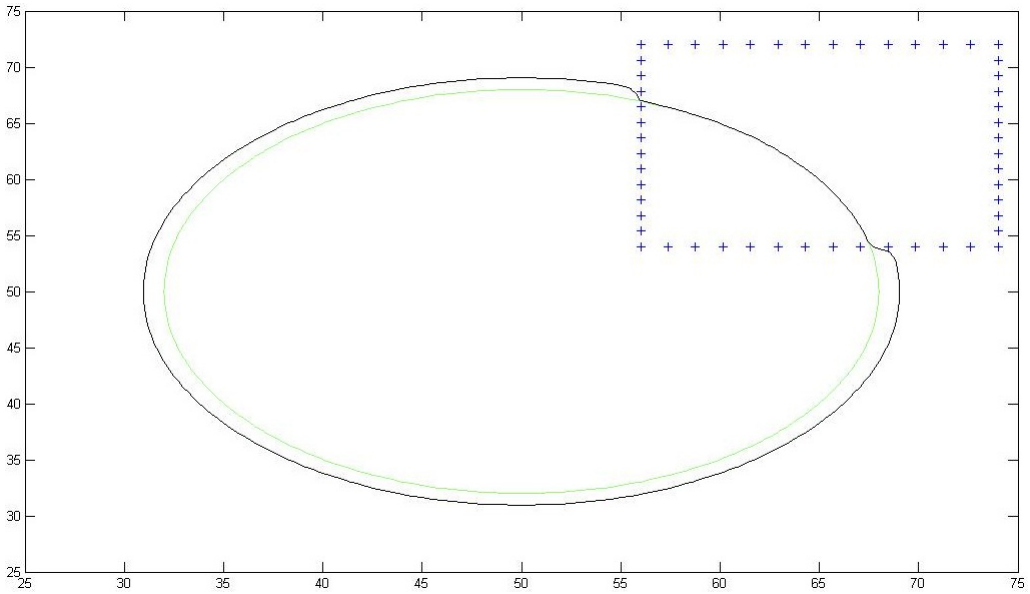


Figura 6.46: $\Delta t = 0.01$, con $||\nabla(\phi)||$ y reinicialización cada 2 iteraciones.

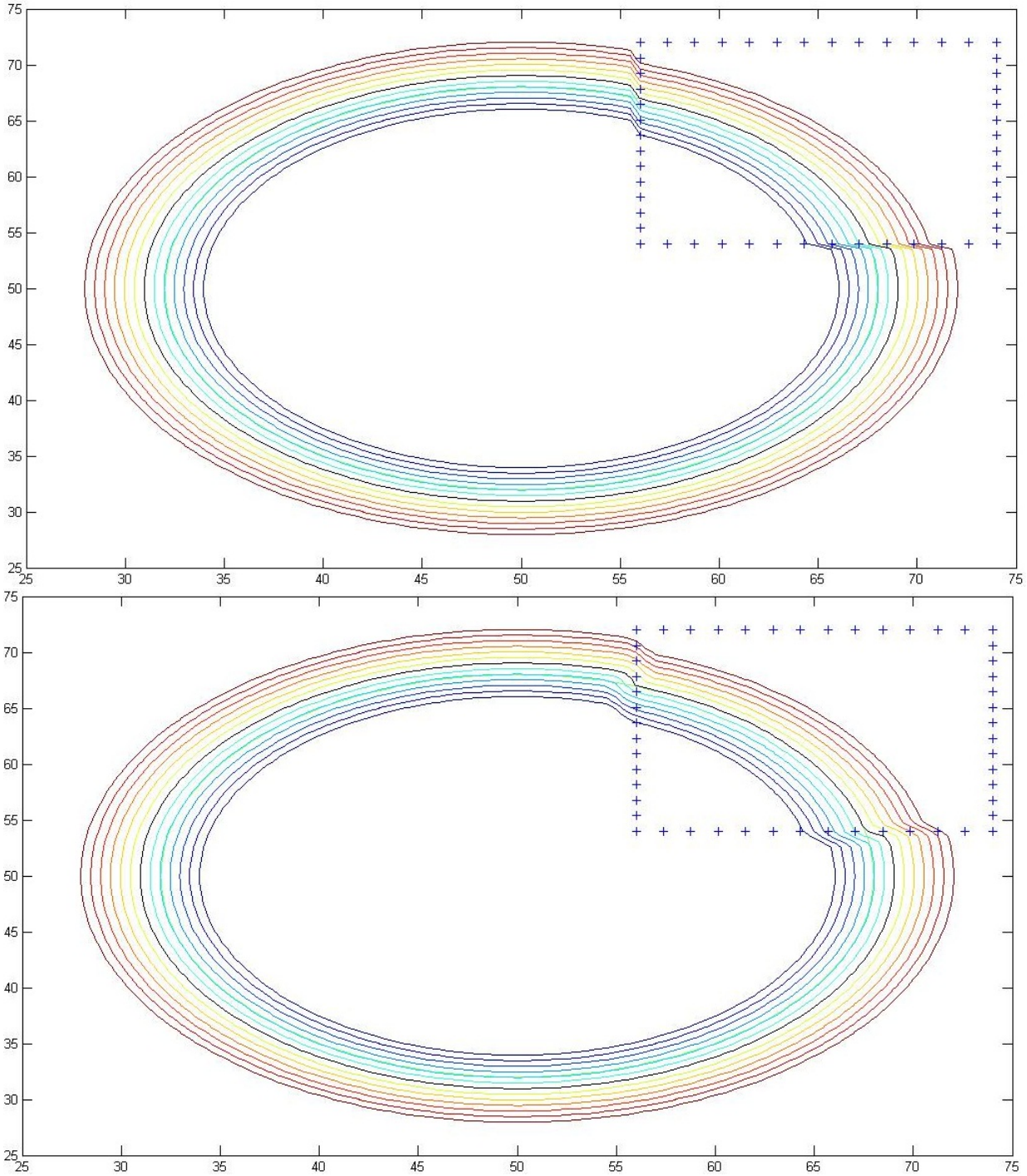


Figura 6.47: (*arriba*) sin reinicialización.(*abajo*) con reinicialización cada 2 iteraciones.

En el caso cuando el material deformable es un cuadrado, la compensación del volumen, hace que las puntas en la interface final (línea sólida negra) hace que se vayan redondeando como se puede ver en la Figura.

	No. de reinicializaciones	Tiempo de reinicialización	Iteraciones	Tiempo total	$\lvert Vol_{\Gamma^c}(\phi) - c\rvert$
Figura 6.48	0	0	6	53 s.	1.75
Figura 6.49	3	89.15 s.	6	145 s.	1.75

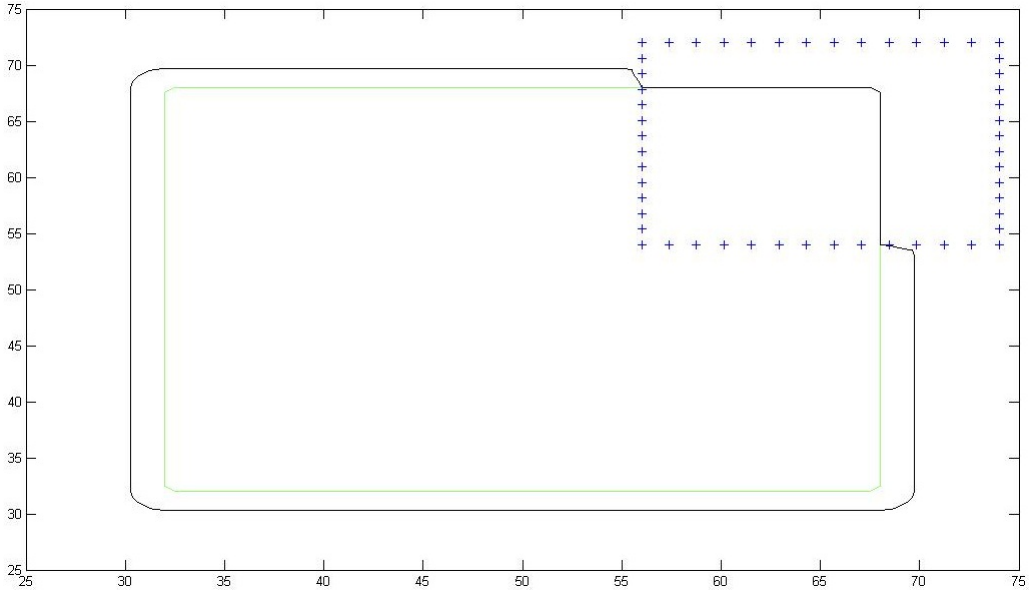


Figura 6.48: $\Delta t = 0.01$, con $||\nabla(\phi)||$ sin reinicialización.

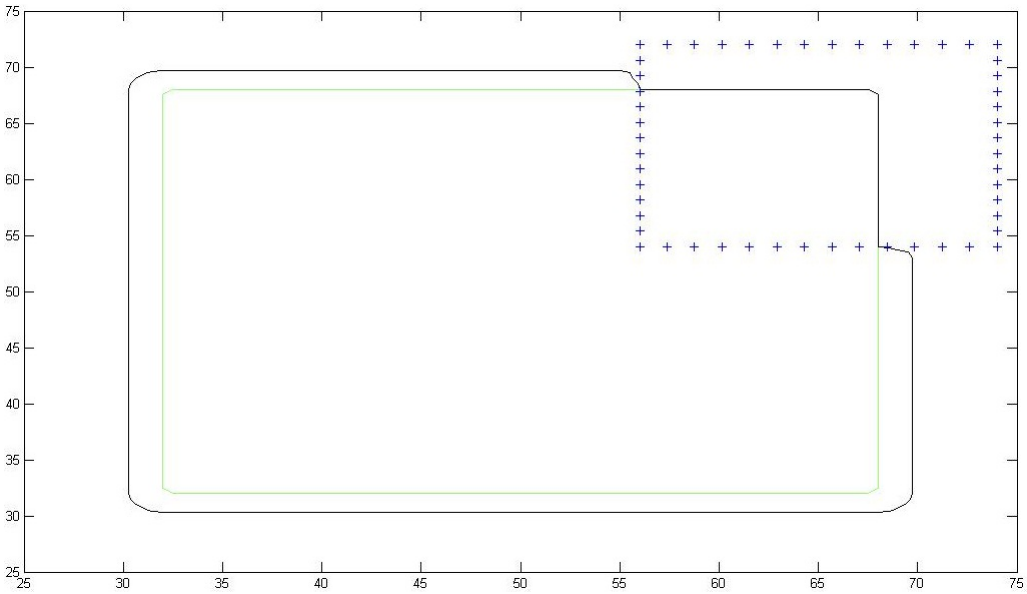


Figura 6.49: $\Delta t = 0.01$, con $||\nabla(\phi)||$ y reinicialización cada 2 iteraciones.

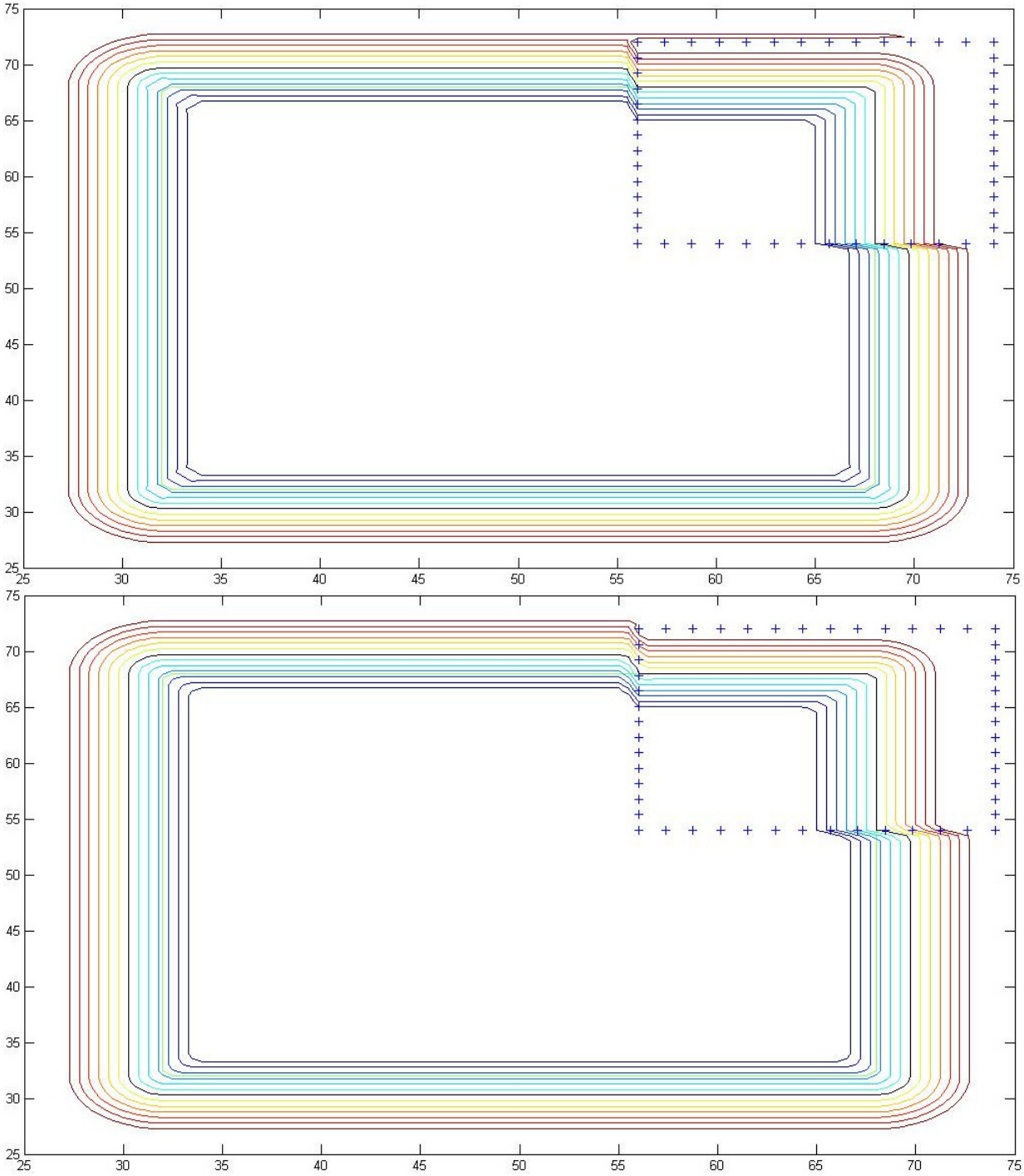


Figura 6.50: (*arriba*) sin reinicialización.(*abajo*) con reinicialización cada 2 iteraciones.

Aunque la reinicialización sea bastante costosa en tiempo, hay casos en los que realmente se necesitan, tal como en el siguiente. Aquí esta opción de reconstruir la función distancia sí ayuda bastante para que sea más estable la evolución de la interface, por lo que no se puede prescindir completamente de esta herramienta.

	No. de reinicializaciones	Tiempo de reinicialización	Iteraciones	Tiempo total	$\lvert Vol_{\Gamma^c}(\phi) - c\rvert$
Figura 6.51	0	0	18	126.3 s.	30.5
Figura 6.52	4	137 s.	8	213 s.	28.5

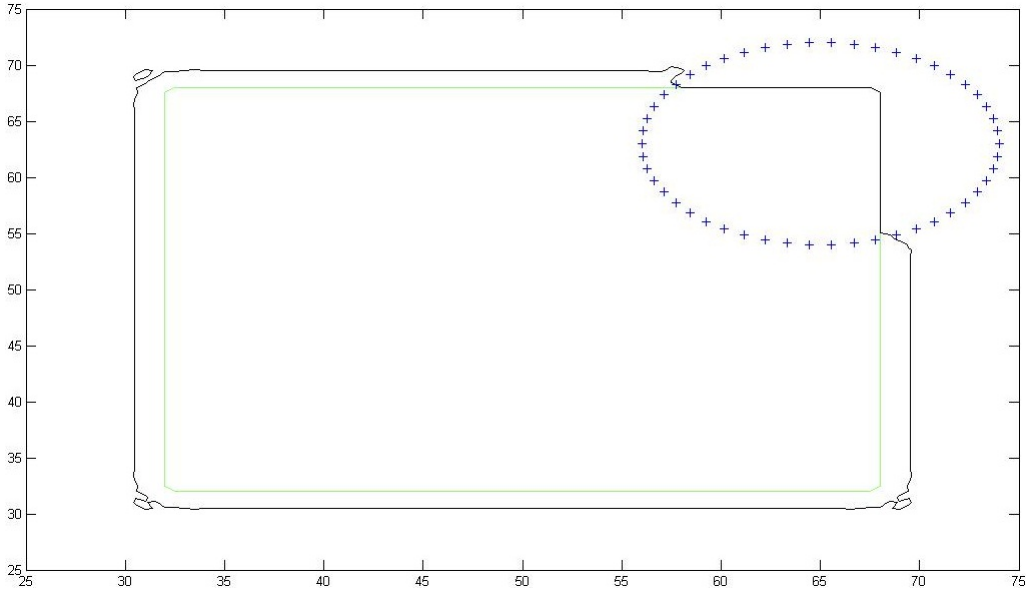


Figura 6.51: $\Delta t = 0.01$, con $||\nabla(\phi)||$ sin reinicialización.

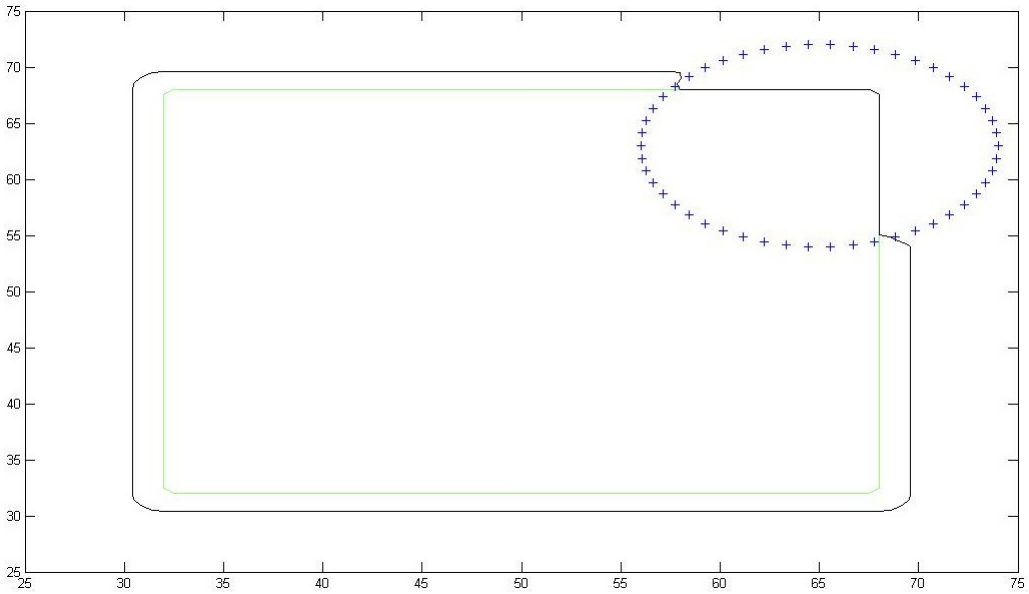


Figura 6.52: $\Delta t = 0.01$, con $||\nabla(\phi)||$ y reinicialización cada 2 iteraciones.

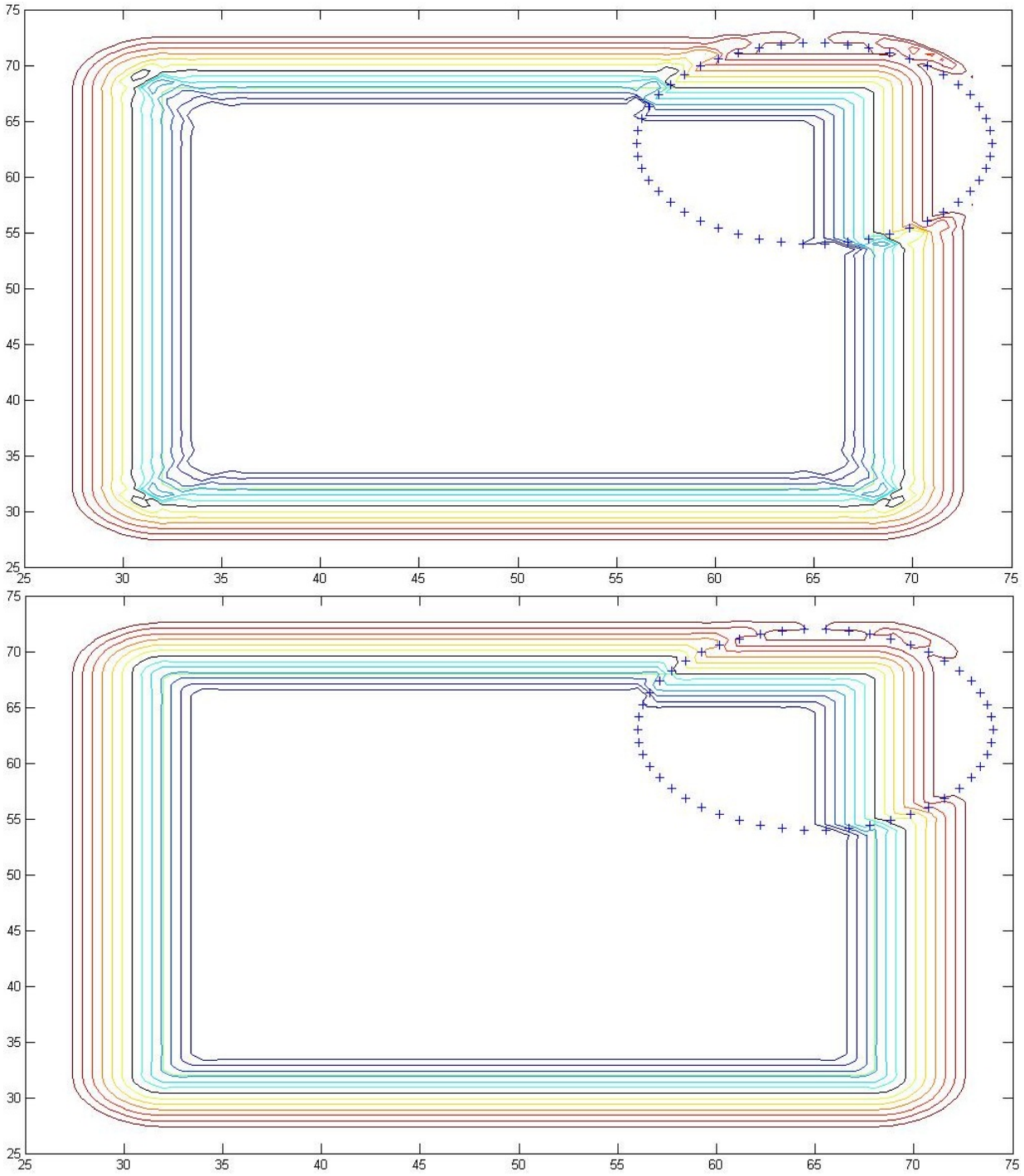


Figura 6.53: (*arriba*) sin reinicialización.(*abajo*) con reinicialización cada 2 iteraciones.

La compensación de volumen se comportó bien de manera general con el valor Δt =0.1, sólo el último caso fue un poco patológico. Aunque la diferencia entre la cantidad de material inicial (volumen o área) del objeto deformable y la última cantidad calculada de éste no llegó a ser completamente cero, un error menor que 1 es bueno. La reinicialización ayudó algunas veces y no mejoró gran cosa en la mayoría de los

casos, pero sí se volvió muy costosa su aplicación, sobre todo al recurrirse frecuentemente en el algoritmo. Es probable que no se pueda prescindir completamente de ella, pero sí debe hacerse una gestión apropiada en su aplicación. La implentación de la deformación local y la compensación de volumen al superponerse funcionan tal cual como se mostró anteriormente, sin embargo aquí se mostró de manera independiente, por lo cual no hay ningún inconveniente.

Hasta aquí llega el nálisis que corresponde a los objetivos de la tesis, pero hay muchas posibilidades para ir avanzando con este problema en el futuro, como la implementación de mejores formas de discretizar las ecuaciones de evolución (5.6) y (5.9), así como otras maneras de evolucionar y deformar la interface, y la aplicación de conceptos físicos, entre otros.

Apéndice A

Algoritmos

A.1. Función general

```
function [ph]=
Ambiente_total(n,dt,k,c1,c2,c,cf,r1,r,vel,formaMD,formaH,rig,reinicia,accion,ecuacion)
%Este programa crea un ambiente virtual: la herramienta virtual (con una representación
%explícita y sus coordenadas están dados por px y py) y el objeto que se deformará (a
%través de una función implícita ph), asemás realizará lo que el usuario desee que
%haga: deformar localmente al objeto o que evolucione de tal forma que compense su
%contenido o volumen, o puede hacer ambos al mismo tiempo.
%Entradas:
    %n=número de puntos de los ejes coordenados
    %dt=paso de tiempo
    %k=número de puntos discretos de la herramienta
    %c1,c2=coordenadas 'x' y 'y' respectivamente, de los centros de masa de los objetos
    %deformables (cada valor puede ser un vector de valores o un valor simplemente);
    %observación:todos los objetos tienen que ser de la misma forma
    %c=punto (vector) correspondiente al centro de masa de la herramienta virtual
    %cf=Punto final de la herramienta virtual
    %r1=dimensiones del material deformable (un valor para el círculo: su radio 'r', o
    %un vector para el rectángulo: su ancho y su alto; también puede ser un vector
    %de radios en el caso de que todos los objetos sean círculos y un vector de
    %anchura y alturas alternados en el caso de los rectángulos),
    %r=dimensiones de la herramienta virtual (un valor para el círculo: su radio 'r',
    %o un vector para el rectángulo: su ancho y su alto)
    %vel=velocidad de la herramienta
```

```
    %formaMD=determina la forma del material deformable ('CIR'=círculo,
    %'REC'=rectángulo)
    %formaH=forma de la herramienta('CIR'=forma de círculo,'REC'=forma de rectúngulo)
    %rig=Índice de rigidez del objeto de deformación en caso de respuesta de colisión
    %reinicia=es la frecuencia que se requiere utilizar la reinicialización de la
    %función distancia con signo ('0'=sin reinicialización,'n' donde n es un número
    %positivo, indica que se reinicializará en cada n iteraciones)
    %accion=lo que se pretende que haga el programa('1'=deformación local de la
    %herramienta,'2'=compensación de volumen,'3'=ambas a la vez,'4'=renderización
    %háptica,'5'=todos a la vez)
    %ecuacion=indica el tipo de ecuación en la deformación por compensación de material
    %(si en el valor anterior es '1', no importa lo que se ponga, pero si es '2' ó '3',
    %entonces se debe escoger entre '1'=que utiliza la función delta o '2'=que utiliza
    %la norma del gradiente)
%Salidas:
    %Función ph

%----------------------------------------
%EJES COORDENADOS
%Se construye la malla cartesiana bidimensional, construyendo dos mallas
%unidimensionales
x0=0; %Punto inicial (eje x)
xn=100; %Punto final (eje x)
y0=0; %Punto inicial (eje y)
yn=100; %Punto final (eje y)
h=(xn-x0)/n; %Tamaño de paso espacial
%Construcción discreta de los ejes X y Y
for i=1:n+1
    X(i)=x0+(i-1)*h;
    Y(i)=y0+(i-1)*h;
end
%-------------------------------------
%MATERIAL DEFORMABLE
%Se construye el material deformable
switch formaMD%Se elige la forma (circunferencia o rectángulo)
     %---------------------------------------
```

```
     %circunferencia
     %Construcción de la circunferencia
     case 'CIR'
          for i=1:size(X,2)
               for j=1:size(Y,2)
                   if size(c1,2)==1
            ph(j,i)=sqrt((X(i)-c1)^2+(Y(j)-c2)^2)-r1;
                   elseif size(c1,2)>1
                        ph(j,i)=sqrt((X(i)-c1(1))^2+(Y(j)-c2(2))^2)-r1(1);
                        for k=2:size(c1,2)
                             ph1(j,i)=sqrt((X(i)-c1(k))^2+(Y(j)-c2(k))^2)-r1(k);
                             ph(j,i)=min(ph(j,i),ph1(j,i));
                        end
                   end
              end
          end
     %----------------------------------------
     %Rectángulo
     %Construcción del rectángulo
     case 'REC'
          for i=1:size(X,2)
               for j=1:size(Y,2)
                   if size(c1,2)==1%El vector r1 debe ser del doble de tamaño de c1 y
%c2
                        if (c1-r1(1)<=X(i))&&(X(i)<=c1+r1(1))&&(c2-r1(2)<=Y(j))&&(Y(j)
<=c2+r1(2))
                            ph(i,j)=max(abs(X(i)-c1)-r1(1),abs(Y(j)-c2)-r1(2));
                 elseif (c1-r1(1)<=X(i))&&(X(i)<=c1+r1(1))&&((c2-r1(2)>Y(j))||
(Y(j)>c2+r1(2)))
                             ph(i,j)=max(abs(X(i)-c1)-r1(1),abs(Y(j)-c2)-r1(2));
                 elseif ((c1-r1(1)>X(i))||(X(i)>c1+r1(1)))&&(c2-r1(2)<=Y(j))&&
(Y(j)<=c2+r1(2))
                             ph(i,j)=max(abs(X(i)-c1)-r1(1),abs(Y(j)-c2)-r1(2));
                        else
                             ph(i,j)=norm([abs(X(i)-c1)-r1(1),abs(Y(j)-c2)-r1(2)]);
                        end
```

```
               elseif size(c1,2)>1%El vector r1 debe ser del doble de tamaño de c1
%y c2
                        if (c1(1)-r1(1)<=X(i))&&(X(i)<=c1(1)+r1(1))&&(c2(1)-r1(2)
<=Y(j))&&(Y(j)<=c2(1)+r1(2))
                             ph(i,j)=max(abs(X(i)-c1(1))-r1(1),abs(Y(j)-c2(1))-r1(2));
                 elseif (c1(1)-r1(1)<=X(i))&&(X(i)<=c1(1)+r1(1))&&((c2(1)-r1(2)>
Y(j))||(Y(j)>c2(1)+r1(2)))
                             ph(i,j)=max(abs(X(i)-c1(1))-r1(1),abs(Y(j)-c2(1))-r1(2));
                 elseif ((c1(1)-r1(1)>X(i))||(X(i)>c1(1)+r1(1)))&&(c2(1)-r1(2)
<=Y(j))&&(Y(j)<=c2(1)+r1(2))
                             ph(i,j)=max(abs(X(i)-c1(1))-r1(1),abs(Y(j)-c2(1))-r1(2));
                        else
                             ph(i,j)=norm([abs(X(i)-c1(1))-r1(1),
abs(Y(j)-c2(1))-r1(2)]);
                        end
                        for k=2:size(c1,2)
                             if (c1(k)-r1(2*k-1)<=X(i))&&(X(i)<=c1(k)+r1(2*k-1))&&
(c2(k)-r1(2*k)<=Y(j))&&(Y(j)<=c2(k)+r1(2*k))
                                  ph1(i,j)=max(abs(X(i)-c1(k))-r1(2*k-1),
abs(Y(j)-c2(k))-r1(2*k));
                  elseif (c1(k)-r1(2*k-1)<=X(i))&&(X(i)<=c1(k)+r1(2*k-1))&&
((c2(k)-r1(2*k)>Y(j))||(Y(j)>c2(k)+r1(2*k)))
                                  ph1(i,j)=max(abs(X(i)-c1(k))-r1(2*k-1),
abs(Y(j)-c2(k))-r1(2*k));
                  elseif ((c1(k)-r1(2*k-1)>X(i))||(X(i)>c1(k)+r1(2*k-1)))&&
(c2(k)-r1(2*k)<=Y(j))&&(Y(j)<=c2(k)+r1(2*k))
                                  ph1(i,j)=max(abs(X(i)-c1(k))-r1(2*k-1),
abs(Y(j)-c2(k))-r1(2*k));
                             else
                                  ph1(i,j)=norm([abs(X(i)-c1(k))-r1(2*k-1),
abs(Y(j)-c2(k))-r1(2*k)]);
                             end
                             ph(j,i)=min(ph(j,i),ph1(j,i));
                        end
                    end
               end
```

```
          end
end
%---------------------------------------
%HERRAMIENTA
%Construcció de la herramienta
switch formaH%Se elige la forma (circunferencia o rectángulo)
     %---------------------------------------
     %Circunferencia
     %Construcción de la circunferencia
     case 'CIR'
          for i=0:k
               px(i+1)=r*cos(2*pi*i/k)+c(1);
               py(i+1)=r*sin(2*pi*i/k)+c(2);
          end
     %---------------------------------------
     %Rectángulo
     %Construcción del rectángulo
     case 'REC'
         k1=ceil(k/4);
         k=4*k1;
         for t=1:k1
             px(t)=c(1)-r(1)+2*(t-1)*r(1)/k1;
             py(t)=c(2)+r(2);
             px(t+k1+1)=c(1)+r(1);
             py(t+k1+1)=c(2)+r(2)-2*(t-1)*r(2)/k1;
             px(t+2*k1+1)=c(1)+r(1)-2*(t-1)*r(1)/k1;
             py(t+2*k1+1)=c(2)-r(2);
             px(t+3*k1+1)=c(1)-r(1);
             py(t+3*k1+1)=c(2)-r(2)+2*(t-1)*r(2)/k1;
         end
end
%-----------------------------------
%FUNCIONES DE DEFORMACIÓN LOCAL Y COMPENSACIÓN DE VOLUMEN
if accion==1%Deformación local
    [ph]=objdef(ph,X,Y,px,py,n,h,dt,k,c,r,vel,formaH,reinicia,accion,0);
elseif accion==2%Compensación volumen material
```

```
    [ph]=objdef(ph,X,Y,px,py,n,h,dt,k,c,r,vel,formaH,reinicia,accion,ecuacion);
elseif accion==3%Incluye a los dos anteriores
    [ph]=objdef2(ph,X,Y,px,py,n,h,dt,k,c,r,vel,formaH,reinicia,ecuacion)
elseif==4%Movimiento de herramienta y renderización háptica
    [ph]=Movlib(ph,X,Y,px,py,n,h,k,rig,accion);
elseif==5%Renderización háptica y deformación de herramienta
    while (c(1)~=cf(1))&&(c(2)~=cf(2))
    [ph,c]=Movlib(ph,X,Y,px,py,n,h,dt,k,formaH,c,cf,r3,rig,accion);
    [ph]=objdef2(ph,X,Y,px,py,n,h,dt,k,c,r,vel,formaH,reinicia,ecuacion);
    end
end
```

A.2. Movimiento de la herramienta

```
function [ph,c]=Movlib(ph,X,Y,px,py,n,h,dt,k,c,cf,r,vel,formaH,rig,accion)
%Esta función mueve a la herramienta a una velocidad vel hasta que haya alguna colisión
%Entradas:
    %ph=La función ph
    %X,Y=Puntos discretos de los ejes X y Y
    %px,py=Vectores de coordenadas X y Y de los puntos de la herramienta
    %n=número de puntos de los ejes coordenados
    %h=Tamaño de paso espacial
    %dt=paso de tiempo
    %k=número de puntos discretos de la herramienta, accion=lo que se pretende que haga
    %el programa('1'=deformación local de la herramienta,'2'=compensación de volumen,
    %'3'=ambas a la vez)
    %c=punto (vector) correspondiente al centro de masa de la herramienta virtual
    %cf=Punto final de la herramienta virtual
    %r=dimensiones de la herramienta virtual (un valor para el cáculo: su radio 'r',
    %o un vector para el rectángulo: su ancho y su alto)
    %vel=velocidad de deformación de la herramienta
    %formaH=forma de la herramienta('CIR'=forma de círculo,'REC'=forma de rectángulo)
    %rig=Índice de rigidez
    %accion=lo que se pretende que haga el programa(en este caso '5'=renderización
    %háptica o '6'=renderización háptica y deformación a la vez)
%Salidas:
```

```
    %ph=La función ph
    %cf=Punto final de la herramienta virtual

%-------------------------------------
%Se guarda elpunto inicial
c0=c;
%Contador de iteraciones antes de colisión
contador=0;
[Xb,Yb,I,J,dinI,dinJ,hd,Ib,mx,my,valorHph,colision]=
rendhap(ph,X,Y,px,py,n,h,k,0,0,0,0,0,0,0,colision,contador,rig,accion);
while colision==0
    %-------------------------------------
    %MOVIMIENTO LIBRE DE LA HERRAMIENTA (ACTUALIZACIÓN DEL CENTRO DE MASAS
    %DE LA HERRAMIENTA).
    %La trayectoria se implementa a partir del centro de
    %masas de la herramienta
    %Posición inicial+(distancia a recorrer=velocidad*tiempo)[dirección de
    %recorrido y magnitud de velocidad]
    c=c+dt*vel*(cf-c0)/norm(c0-cf);
    %Actualización de la posición del centro de masa de la herramienta
    %----------------------------------------
    %HERRAMIENTA
    %Construcción de la herramienta
    switch formaH%Se elige la forma (circunferencia o rectángulo)
    %----------------------------------------
    %Circunferencia
    %Construcción de la circunferencia
    case 'CIR'
        for i=0:k
        px(i+1)=r*cos(2*pi*i/k)+c(1);
            py(i+1)=r*sin(2*pi*i/k)+c(2);
        end
    %----------------------------------------
    %Rectángulo
    %Construcción del rectángulo
    case 'REC'
```

```
        k1=ceil(k/4);
        k=4*k1;
        for t=1:k1
            px(t)=c(1)-r(1)+2*(t-1)*r(1)/k1;
            py(t)=c(2)+r(2);
            px(t+k1+1)=c(1)+r(1);
            py(t+k1+1)=c(2)+r(2)-2*(t-1)*r(2)/k1;
            px(t+2*k1+1)=c(1)+r(1)-2*(t-1)*r(1)/k1;
            py(t+2*k1+1)=c(2)-r(2);
            px(t+3*k1+1)=c(1)-r(1);
            py(t+3*k1+1)=c(2)-r(2)+2*(t-1)*r(2)/k1;
        end
    end
    %-------------------------------------
    %RENDERIZACIÓN HÁPTICA
    %Se actualiza contador
    contador=contador+1;
    %Se llama a la función de rendereización háptica
    [Xb,Yb,I,J,dinI,dinJ,hd,Ib,mx,my,valorHph,colision]=
rendhap(ph,X,Y,px,py,n,h,k,I,J,dinI,dinJ,hd,mx,my,colision,contador,rig,accion);
    %-------------------------------------
    %VISUALIZACIÓN DE MOVIMIENTO DE LA HERRAMIENTA
    %Material deformable
    contour(X,Y,ph',[0,0])
    hold on
    %Rectángulo envolvente de la herramienta
    line([X(I(1)),X(I(2))],[Y(J(1)),Y(J(1))])
    line([X(I(2)),X(I(2))],[Y(J(1)),Y(J(2))])
    line([X(I(1)),X(I(2))],[Y(J(2)),Y(J(2))])
    line([X(I(1)),X(I(1))],[Y(J(2)),Y(J(1))])
    %Herramienta
    plot(px,py,'x')
    drawnow;
    hold off
end
```

A.3. Deformación de interface

```
%FUNCIÓN DE DEFORMACIÓN (O DEFORMACIÓN LOCAL O COMPENSACIÓN DE VOLUMEN)
function [ph]=objdef(ph,X,Y,px,py,n,h,dt,k,c,r,vel,formaH,reinicia,accion,ecuacion)
%Esta función se encarga de deformar el material tanto localmente así como para
%compensar el material o el espacio que ha 'perdido' debido a la deformación local.
%Entradas:
    %ph=La función ph
    %X,Y=Puntos discretos de los ejes X y Y
    %px,py=Vectores de coordenadas X y Y de los puntos de la herramienta
    %n=número de puntos de los ejes coordenados
    %h=Tamaño de paso espacial
    %dt=paso de tiempo
    %k=número de puntos discretos de la herramienta, accion=lo que se pretende que
    %haga el programa('1'=deformación local de la herramienta,'2'=compensación de
    %volumen,'3'=ambas a la vez)
    %c=punto (vector) correspondiente al centro de masa de la herramienta virtual
    %r=dimensiones de la herramienta virtual (un valor para el círculo: su radio 'r',
    %o un vector para el rectángulo: su ancho y su alto)
    %vel=velocidad de la herramienta
    %formaH=forma de la herramienta('CIR'=forma de círculo,'REC'=forma de rectángulo)
    %reinicia=es la frecuencia que se requiere utilizar la reinicialización de la
    %función distancia con signo ('0'=sin reinicialización,'n' donde n es un número
    %positivo, indica que se reinicializa en cada n iteraciones)
    %accion=lo que se pretende que haga el programa('1'=deformación local de la
    %herramienta,'2'=compensación de volumen,'3'=ambas a la vez)
    %ecuacion=indica el tipo de ecuación en la deformación por compensación de material
    %(si en el valor anterior es '1', no importa lo que se ponga, pero si es '2' ó '3',
    %entonces se debe escoger entre '1'=que utiliza la función delta o '2'=que utiliza
    %la norma del gradiente)
%Salidas:
    %ph=Función ph

%-------------------------------------
%Se guarda el valor de la función actual
ph0=ph;
%-------------------------------------
```

```
%PARÁMETROS
%Parámetro que detendrá el ciclo de deformación
paro=0;%Inicialización
%-----------------------------------
%ACCIÓN DE DEFORMACIÓN LOCAL
if accion==1
    %Indicador de detección de colisión (=cero (sin colisión), no nulo (colisión)),
    %para llevar a cabo la deformación local mientras haya colisión entre la
    %herramienta y el objeto que se deforma
    colision=0;%Inicialización
    %-----------------------------------
    %ZONA DE INFLUENCIA DE LA HERRAMIENTA
    %Malla que sirve para definir la zona de influencia de la herramienta
    ph1=zeros(n+1,n+1);%Inicialización
    %Se define la zona de influencia de la herramienta
    for i=1:size(X,2)
        for j=1:size(Y,2)
            switch formaH%Se elige la forma (circunferencia o rectágulo)
                case 'CIR'
                    %Circunferencia
                    %Construcción de lacircunferencia
                    r3=sqrt((X(i)-c(1))^2+(Y(j)-c(2))^2);
                    if r3<=r
                        ph1(i,j)=1;
                    end
                case 'REC'
                    %Rectángulo
                    %Construcción del rectángulo
                    if (c(1)-r(1)<=X(i))&&(X(i)<=c(1)+r(1))&&(c(2)-r(2)<=Y(j))&&
(Y(j)<=c(2)+r(2))
                        ph1(i,j)=1;
                    end
            end
        end
    end
    %-----------------------------------
```

```
    %CICLO DE DEFORMACIÓN LOCAL
    while paro~=1
        %Se actualiza contador
        contador=contador+1;
        %Se calcula la norma del gradiente
        grad=normgrad(ph,n,h);
        %Se llama a la función de rendereización háptica para utilizar la parte de
    %detección de colisión
        [Xb,Yb,I,J,dinI,dinJ,hd,Ib,mx,my,valorHph,colision]=
rendhap(ph,X,Y,px,py,n,h,k,0,0,0,0,0,0,0,colision,contador,0,0);
    %-------------------------------------
        %DEFORMACIÓN LOCAL
        %Si todavÍa hay contacto de la herramienta con el material deformable.
        if (colision~=0)||(itera>=contador)
        %EcuaciÓn de deformación local
            ph=ph+dt*vel*grad.*ph1;
            colision=0;%Se inicializa para que no afecte a los otros ciclos
        else
            paro=paro+1;
        end
        %-------------------------------------
        %REINICIALIZACIÓN
        if reinicia~=0
            if rem(contador,reinicia)==0
                [ph] = reinit_SD(ph, h, h, 0.9, 'WENO', 1);
            end
        end
        %-------------------------------------
        %VISUALIZACIÓN
    %Objeto deformable inicial
        contour(X,Y,ph0',[0,0])
        hold on
     %Conjuntos de nivel
        contour(X,Y,ph',[-2:0.2:2])
     %Objeto deformable
        contour(X,Y,ph',[0,0],'k')
```

```
     %Herramienta
        plot(px,py,'o')
        drawnow
        hold off
    end
elseif accion==2
    %-------------------------------------
    %ZONA EXTERIOR DE LA HERRAMIENTA
    %Malla que sirve para definir la zona exterior de la herramienta, la función chi
    %del exterior
    ph2=zeros(n+1,n+1)+1;
    %-------------------------------------
    %VOLUMEN Y ZONA DE INFLUENCIA DE LA HERRAMIENTA
    %Volumen inicial
    Vol0=0;%Inicialización
    for i=1:size(X,2)
        for j=1:size(Y,2)
            %-------------------------------------
            %Se comienza a calcular el volumen inicial
            if ph(i,j)<=0
                Vol0=Vol0+1;
            end
            %-------------------------------------
            %Se define la zona de influencia de la herramientand
            switch formaH%Se elige la forma (circunferencia o rectángulo)
                case 'CIR'
                    %Circunferencia
                    r3=sqrt((X(i)-c(1))^2+(Y(j)-c(2))^2);
                    if r3<=r
                        ph2(i,j)=0;
                    end
                case 'REC'
                    %Rectángulo
                    if (c(1)-r(1)<=X(i))&&(X(i)<=c(1)+r(1))&&(c(2)-r(2)<=Y(j))&&
(Y(j)<=c(2)+r(2))
                        ph2(i,j)=0;
```

```
                end
            end
        end
    end
%Se termina de calcular el volumen inicial
Vol0=Vol0*h*h;
%-----------------------------------
%CICLO DE COMPENSACIÓN DE VOLUMEN
while paro~=1
    %Se actualiza contador
    contador=contador+1;
    %Se calcula la norma del gradiente
    grad=normgrad(ph,n,h);
%Se calcula el nuevo volumen interior
    Vol=Volg(ph,ph2,n,h);%Se calcula el nuevo volumen
    %Diferencia entre el volumen actual y el inicial
dif=(Vol-Vol0)
    %-----------------------------------
    %MOVIMIENTO PARA LA COMPENSACIÓN DE VOLUMEN
    if (dif~=0)&&(ecuacion==1)
        %Opción 1)Se utiliza la función delta
        ph=ph+dt*dif*dlcon(ph,n,h).*ph2;
    elseif (dif~=0)&&(ecuacion==2)
        %Opción 2)Se utiliza la norma del gradiente
        ph=ph+dt*dif*grad.*ph2;
    else
        paro=paro+1;
    end
%-----------------------------------
    %REINICIALIZACIÓN
    if reinicia~=0
        Treinicia=cputime;
        if rem(contador,reinicia)==0
            [ph] = reinit_SD(ph, h, h, 0.9, 'WENO', 1);
        end
    end
```

```
        %-----------------------------------
        %VISUALIZACIÓN
    %Objeto deformable inicial
        contour(X,Y,ph0',[0,0])
        hold on
     %Conjuntos de nivel
        contour(X,Y,ph',[-2:0.2:2])
     %Objeto deformable
        contour(X,Y,ph',[0,0],'k')
     %Herramienta
        plot(px,py,'o')
        drawnow
        hold off
    end
end

%FUNCIÓN DE DEFORMACIÓN2 (DEFORMACIÓN LOCAL Y COMPENSACIÓN DE VOLUMEN)
function [ph]=objdef2(ph,X,Y,px,py,n,h,dt,k,c,r,vel,formaH,reinicia,ecuacion)
%Esta función se encarga de deformar el material tanto localmente así como para
%compensar el material o el espacio que ha 'perdido' debido a la deformación local.
%Entradas:
    %ph=La función ph
    %X,Y=Puntos discretos de los ejes X y Y
    %px,py=Vectores de coordenadas X y Y de los puntos de la herramienta
    %n=Número de puntos de los ejes coordenados
    %h=Tamaño de paso espacial
    %dt=Tamaño de paso de tiempo
    %k=Número de puntos discretos de la herramienta
    %c=Punto (vector) correspondiente al centro de masa de la herramienta virtual
    %r=Dimensiones de la herramienta virtual (un valor para el cálculo: su radio 'r',
    %o un vector para el rectángulo: su ancho y su alto)
    %vel=Velocidad de la herramienta
    %formaH=Forma de la herramienta('CIR'=forma de círculo,'REC'=forma de rectángulo)
    %reinicia=Es la frecuencia que se requiere utilizar la reinicialización de la
    %función distancia con signo ('0'=sin reinicialización,'n' donde n es un número
    %positivo, indica que se reinicializará en cada n iteraciones)
```

```
    %ecuacion=Indica el tipo de ecuación en la deformación por compensación de material
    %(si en el valor anterior es '1', no importa lo que se ponga, pero si es '2' ó '3',
    %entonces se debe escoger entre '1'=que utiliza la función delta o '2'=que
    %utiliza la norma del gradiente)
%Salidas:
    %ph=Función ph

%-------------------------------------
%Se guarda la función actual
ph0=ph;
%-------------------------------------
%ZONA DE INFLUENCIA DE LA HERRAMIENTA
%Malla que sirve para definir la zona de influencia de la herramienta
ph1=zeros(n+1,n+1);%Se inicializa
%-------------------------------------
%ZONA EXTERIOR DE LA HERRAMIENTA
%Malla que sirve para definir la zona exterior de la herramienta, la función chi
%del exterior
ph2=zeros(n+1,n+1)+1;%Se inicializa
%-------------------------------------
%VOLUMEN Y ZONA DE INFLUENCIA DE LA HERRAMIENTA
%Volumen inicial
Vol0=0;%Inicialización
for i=1:size(X,2)
    for j=1:size(Y,2)
        %-------------------------------------
        %Se comienza a calcular el volumen inicial
        if ph(i,j)<=0
             Vol0=Vol0+1;
        end
        %-------------------------------------
        %Se define la zona de influencia de la herramienta
        %Circunferencia
        %Construcción de lacircunferencia
        switch formaH%Se elige la forma (circunferencia o rectángulo)
             case 'CIR'
```

```
                r3=sqrt((X(i)-c(1))^2+(Y(j)-c(2))^2);
                if r3<=r
                    ph1(i,j)=1;
                    ph2(i,j)=0;
                end
            %-------------------------------------
            %Rectángulo
            %Construcción del rectÃ¡ngulo
            case 'REC'
                if (c(1)-r(1)<=X(i))&&(X(i)<=c(1)+r(1))&&(c(2)-r(2)<=Y(j))&&
(Y(j)<=c(2)+r(2))
                    ph1(i,j)=1;
                    ph2(i,j)=0;
                end
        end
    end
%Se termina de calcular el volumen inicial
Vol0=Vol0*h*h;
%-------------------------------------
%Variable que detiene el ciclo de deformación
paro=0;%Inicialización
%Indocador de colisión ('0'=No hay colisión, '~=0'=Sí hay colisión)
colision=0;%Inicialización
%-------------------------------------
%CICLO DE DEFORMACIÓN
while paro~=2
    %Se actualiza contador
    contador=contador+1;
    %Se calcula la norma del gradiente
    grad=normgrad(ph,n,h);
    %Variables que se tienen que inicializar debido a la definición de la función
    %objdef2
    %Se llama a la función de rendereización háptica para utilizar la parte de
    %detección de colisión
    [Xb,Yb,I,J,dinI,dinJ,hd,Ib,mx,my,valorHph,colision]=
rendhap(ph,X,Y,px,py,n,h,k,0,0,0,0,0,0,0,colision,contador,0,0);
```

```
%-------------------------------------
    %DEFORMACIÓN LOCAL
    %Si todavía hay contacto entre la herramienta y el material deformable.
    if (colision~=0)||(itera>=colision)
    %Ecuación de deformación local
        ph=ph+dt*vel*grad.*ph1;
        colision=0;%Se inicializa para que no afecte a los otros ciclos
    else
        paro=paro+1;
    end
    %Se calcula el nuevo gradiente
    grad=normgrad(ph,n,h);
    %Se calcula el nuevo volumen interior
    Vol=Volg(ph,ph2,n,h);%Se calcula el nuevo volumen
    %Diferencia entre el volumen actual y el inicial
    dif=(Vol-Vol0);
    %--------------------------------------
    %MOVIMIENTO PARA LA COMPENSACIÓN DE VOLUMEN
    if (dif~=0)&&(ecuacion==1)
        %Opción 1)Se utiliza sólo la función delta
        ph=ph+dt*(Vol-Vol0)*dlcon(ph,n,h).*ph2;
    elseif (dif~=0)&&(ecuacion==2)
        %Opción 2)Se utiliza la norma del gradiente
        ph=ph+dt*dif*grad.*ph2;
    else
        paro=paro+1;
    end
    %--------------------------------------
    %REINICIALIZACIÓN
    %El valor de reinicia es la frecuencia de reinicialización
    if reinicia~=0
        if rem(contini,reinicia)==0
            [ph] = reinit_SD(ph, h, h, 0.9, 'WENO', 1);
        end
    end
    %--------------------------------------
```

```
    %VISUALIZACIÓN
%Objeto deformable inicial
    contour(X,Y,ph0',[0,0])
    hold on
%Conjuntos de nivel
    contour(X,Y,ph',[-2:0.2:2])
%Objeto deformable
    contour(X,Y,ph',[0,0],'k')
%Herramienta
    plot(px,py,'o')
    drawnow
    hold off
end
```

A.4. Renderización háptica

```
%RENDERIZACIÓN HÁPTICA
[Xb,Yb,I,J,dinI,dinJ,hd,Ib,mx,my,valorHph,colision]=
rendhap(ph,X,Y,px,py,n,h,k,I,J,dinI,dinJ,hd,mx,my,colision,contador,rig,accion);
%Esta función llama a otra que calcula los valores de los puntos discretos de la
%herramienta virtual para determinar se se ha llevado a cabo una colisión con un
%objeto, en dado caso en que sí hubiera, este algortimo llama a una función que se
%encarga de calcular las fuerzas de contacto.
%Entradas:
    %ph=Función ph
    %X,Y=Vectores de los valores de la discretización de lo ejes coordenados, X y Y,
    %respectivamente
    %px,py=Coordenadas x y y, respectivamente de los puntos discretos de la herramienta
    %n=Número de puntos en que se discretiza los ejes coordenados
    %h=Tamaño de paso espacial
    %k=Vector de coordenadas x y y, respectivamente, máximas y mínimas del cuadro que
    %envuelve a la herramienta
    %I,J=Vector de coordenadas x y y, respectivamente, máximas y mínimas del cuadro que
    %envuelve a la herramienta
    %dinI,dinJ=Diferencia entre los índices de las coordenas x y y, respectivamente, de
    %los vértices opuestos correspondiente al rectángulo envolvente
```

```
    %hd=vector de distancias de que hay entre el punto cuya coordenada x es mx, con
    %respecto a los lados verticales de la celda que lo contiene, así como también de
    %las distancias del punto cuya coordenada y es my con respecto a los lados
    %horizontales de la celda que lo contiene
    %mx,my=Coordenadas sobre el eje x y y máximas de los puntos de la herramienta
    %colision=Parámetro indicador de colisión ('0'=No hay colisión, '<=0'=Sí hay
    %colisión)
    %contador=Indica el número de iteraciones totales que se han llevado a cabo antes
    %de haber colisiones
    %mx,my=Coordenadas sobre el eje x y y mínimas de los puntos de la herramienta
    %rig=Índice de de rigidez en caso de respuesta de colisiónan
    %acción1=Indicador que en caso de tener el valor '1' permite calcular las fuerzas
    %de contacto, en otro caso no lo hace
%Salidas:
    %Xb,Yb=Índices o lugares de los puntos de la herramienta que están en colisión
    %dentro del vector Xb
    %I,J=Vector de coordenadas x y y, respectivamente, máximas y mínimas del cuadro
    %que envuelve a la herramienta
    %dinI,dinJ=Diferencia entre los índices de las coordenas x y y, respectivamente, de
    %los vértices opuestos correspondiente al rectángulo envolvente
    %hd=Vector de distancias de que hay entre el punto cuya coordenada x es mx, con
    %respecto a los lados verticales de la celda que lo contiene, así como también de
    %las distancias del punto cuya coordenada y es my con respecto a los lados
    %horizontales de la celda que lo contiene
    %Ib=Índices o lugares de los puntos de la herramienta que están en colisión dentro
    %del vector Xb
    %mx,my=Coordenadas sobre el eje x y y mínimas de los puntos de la herramienta
    %valorHph=Valor de la función phi de los puntos de la herramienta en colisión
    %colision=Parámetro indicador de colisión ('0'=No hay colisión, '<=0'=Sí hay
    %colisión)

%---------------------------------------
%DETECCIÓN DE COLISIÓN
if colision==0
    if contador==0
        %---------------------------------------
```

```
    %Menores coordenadas x e y de la herramienta (para que el rectángulo envolvente
%no tenga mucho margen)
    mx=min(px);%Coordenada mínima x
    s=min(find(px==mx));%Se encuentra el Índice de px donde se encuentra el
    %número más chico. "min" es para asegurar que t sea real y no vector.
    my=min(py);%Coordenada mínima y
    Mx=max(px);%Coordenada máxima en x
    My=max(py); %Coordenada mínima en y
    %-------------------------------------
    %CONSTRUCCIÓN DEL CUADRADO ENVOLVENTE DE LA HERRAMIENTA
    %Esta parte sólo sirve inicialmente.
    for i=1:n
        if (X(i)<=mx)&&(mx<X(i+1))
            I(1)=i;
        end
        if (X(i)<=Mx)&&(Mx<X(i+1))
            I(2)=i+1;.
            dinI=abs(I(2)-I(1));
end
        if (Y(i)<=my)&&(my<Y(i+1))
            J(1)=i;
        end
        if (Y(i)<=My)&&(My<Y(i+1))
            J(2)=i+1;
    dinJ=abs(J(2)-J(1))+1;
        end
    end
    %Se ordenan los valores de las coordenadas de la herramienta de menor a mayor
%con respecto a px
    [Xb,Yb]=orden(px,py,k);
    %Valor mínimo de todas las coordenadas Y de la herramienta
    Ymy=min(Yb);
%Índice correspondiente a la coordenada Ymy en Yb
    n1=min(find(Yb==Ymy));
    %-----------------------------------
    %DETECCIÓN DE COLISIÓN
```

```
        [Pcx,Pcy,hd,Ib,valorHph,colision]=
detcol(ph,X,Y,n,k,I,J,Xb,Yb,mx,my,n1,colision);
elseif contador~=0
        %Se ordenan los valores de las coordenadas de la herramienta de menor a mayor
    %con respecto a px
        [Xb,Yb]=orden(px,py,k);
        %Valor mínimo de todas las coordenadas Y de la herramienta
        Ymy=min(Yb);
    %Índice correspondiente a la coordenada Ymy en Yb
        n1=min(find(Yb==Ymy));
        %-----------------------------------
        %RECONSTRUCCIÓN DEL RECTÁNGULO ENVOLVENTE DE LA HERRAMIENTA
        %Se verifica el desplazamiento de la herramienta, hacia la izquierda,
        %derecha, arriba o abajo, para reconstruir la del cuadrado envolvente
        %en todo el recorrido.
        if (Xb(1)<mx) %Si la herramienta se mueve hacia la izquierda
            if (X(I(1))<Xb(1))&&(Xb(1)<X(I(1)+1))
                I(1)=I(1)-floor(abs((abs(Xb(1)-mx)-hd(2)))/h);
            else
                I(1)=I(1)-floor(abs((abs(Xb(1)-mx)-hd(1)))/h)-1;
            end
            I(2)=I(1)+dinI;
            mx=Xb(1);
        elseif Xb(1)>mx %Si la herramienta se mueve a la derecha
            if (X(I(1))<Xb(1))&&(Xb(1)<X(I(1)+1))
                I(1)=I(1)+floor(abs((abs(Xb(1)-mx)-hd(2)))/h);
            else
                I(1)=I(1)+floor(abs((abs(Xb(1)-mx)-hd(2)))/h)+1;
            end
            I(2)=I(1)+dinI;
            mx=Xb(1);
        end
        if (Ymy<my) %Si la herramienta se mueve hacia abajo
            if (Y(J(1))<Ymy)&&(Ymy<Y(J(1)+1))
                J(1)=J(1)-floor(abs((abs(Ymy-my)-hd(4)))/h);
            else
```

```
                J(1)=J(1)-floor(abs((abs(Ymy-my)-hd(3)))/h)-1;
            end
            J(2)=J(1)+dinJ;
            my=Ymy;
        elseif Ymy>my %Si la herramienta se mueve hacia arriba
            if (Y(J(1))<Ymy)&&(Ymy<Y(J(1)+1))
                J(1)=J(1)+floor(abs((abs(Ymy-my)-hd(4)))/h);
            else
                J(1)=J(1)+floor(abs((abs(Ymy-my)-hd(4)))/h)+1;
            end
            J(2)=J(1)+dinJ;
            my=Ymy;
        end
        %-------------------------------------
        %DETECCIÓN DE COLISIÓN
        [Pcx,Pcy,hd,Ib,valorHph,colision]=
detcol(ph,X,Y,n,k,I,J,Xb,Yb,mx,my,n1,colision);
end
end
%-------------------------------------
%RESPUESTA DE COLISIÓN
if (colision~=0)&&(accion>3)
    [F,gradx,grady,m]=respcol(X,Y,h,n,ph,Xb,Yb,Pcx,Pcy,valorHph,Ib,rig);
end

%DETECCIÓN DE COLISIÓN
function [Pcx,Pcy,hd,Ib,valorHph,colision]=
detcol(ph,X,Y,n,k,I,J,Xb,Yb,mx,my,n1,colision)
%Esta función sirve para calcular el valor ph decada uno de los puntos de la
%herramienta por medio de interpolación de los valores de los puntos de la celda en
%donde se encuentra cada uno
%Entradas:
    %ph=La función ph
    %X,Y=Vectores de los valores de la discretización de lo ejes coordenados, X y Y,
    %respectivamente
    %n=Número de puntos en que se discretiza los ejes coordenados
```

```
    %k=Número de puntos de la herramienta
    %I,J=Vector de coordenadas x y y, respectivamente, máximas y mínimas del cuadro que
    %envuelve a la herramienta
    %Xb,Yb=vectores de coordenadas x y y, respectivamente, de los puntos de la
    %herramienta ordenados de menor a mayor con respecto a los valores del vector pcx
    %mx,my=Coordenadas sobre el eje x y y mínimas de los puntos de la herramienta
    %n1=Índice
    %colision=Variable que indica si ya hubo o no colisión ('0'=no hay colisión,
    %'>0'= ya hubo colisiones)
%Salidas:
    %Pcx,Pcy=Índices de las coordenadas x y y, respectivamente, del vértice extremo
    %derecho inferior de la celda en que se encuentra el punto de al herramienta en
    %colisión
    %hd=vector de distancias de que hay entre el punto cuya coordenada x es mx, con
    %respecto a los lados verticales de la celda que lo contiene, así como también de
    %las distancias del punto cuya coordenada y es my con respecto a los lados
    %horizontales de la celda que lo contiene
    %Ib=Índices o lugares de los puntos de la herramienta que están en colisión dentro
    %del vector Xb
    %valorHph=El valor de la función ph de los puntos de la herramienta en colisión
    %colision=Variable que indica si ya hubo o no colisión ('0'=no hay colisión,
    %'>0'= ya hubo colisiones)

%----------------------------------------
%Se limita el lugar donde puede alcanzar el cuadrado envolvente para aplicar el
%algoritmo
if (1<=J(1))&&(J(1)<n+1)&&(1<J(2))&&(J(2)<=n+1)&&(1<=I(1))&&(I(1)<n+1)&&
(1<I(2))&&(I(2)<=n+1)
    %----------------------------------------
    %Se revisan todas las celdas de todos los puntos de la herramienta barriendo las
    %celdas del cuadro envolvente.
    consi=I(1);
    consj=J(1);
    for t=1:k
        for i=consi:I(2)-1 %Se empieza a verificar desde la celda con la coordenada en
    %x más pequeña.
```

```
            if (X(i)<=Xb(t))&&(Xb(t)<=X(i+1))
                consi=i;
                if t==1
                    hd(1)=abs(X(i)-mx);%distancia con el lado izquierdo de la celda
                    hd(2)=abs(mx-X(i+1));%distancia con el lado derecho de la celda
                end
                break;
            end
        end
        for i=J(1):J(2)-1 %Se empieza a verificar desde la celda con la coordenada en y
    %más pequeña.
            if (Y(i)<=Yb(t))&&(Yb(t)<=Y(i+1))
                consj=i;
                if t==n1
                    hd(3)=abs(Y(i)-my);%distancia con el lado de abajo de la celda
                    hd(4)=abs(my-Y(i+1));%distancia con el lado de la celda
                end
                break;
            end
        end
        %---------------------------------------
        %DETECCIÓN DE COLISIÓN
        %Se interpola el valor de los puntos de la herramienta con los valores de la
    %función distancia de los vértices de la celda que les corresponde
        Hph=(ph(consi,consj)*(X(consi+1)-Xb(t))*(Y(consj+1)-Yb(t))+
ph(consi+1,consj)*(Xb(t)-X(consi))*(Y(consj+1)-Yb(t))+
ph(consi,consj+1)*(X(consi+1)-Xb(t))*(Yb(t)-Y(consj))+
ph(consi+1,consj+1)*(Xb(t)-X(consi))*(Yb(t)-Y(consj)))
/((X(consi+1)-X(consi))*(Y(consj+1)-Y(consj)));
        %Se seleccionan los puntos, valores e índices correspondientes de los puntos
    %que están en colisión
        if Hph<=0
           colision=colision+1;
           valorHph(colision)=Hph;
           Pcx(colision)=consi;
           Pcy(colision)=consj
```

```
            Ib(colision)=t;
        end
    end
    %Variables que se tienen que inicializar debido a la definición de la función
    %detcol
    if colision==0
        valorHph=0;
        Pcx=0;
        Pcy=0;
        Ib=0;
    end
end

%RESPUESTA DE COLISIÓN
function [F,gradx,grady,m]=respcol(ph,X,Y,Pcx,Pcy,n,h,Xb,Yb,Ib,valorHph,rig)
%Esta función se encarga de encontrar los vectores de fuerza correspondientes a cada
%uno de los puntos de herramienta en colisión, utilizando la de Hooke. También calcula
%la fuerza final ejercida en el centro de masas de la herramienta.
%Entradas:
    %ph=Función ph
    %X,Y=Vectores de los valores de la discretización de lo ejes coordenados, X y Y,
    %respectivamente
    %Pcx,Pcy=Índices de las coordenadas x y y, respectivamente, del vértice extremo
    %derecho inferior de la celda en que se encuentra el punto de al herramienta en
    %colisión
    %n=Número de puntos en que se discretiza los ejes coordenados
    %h=Tamaño de paso espacial
    %Xb,Yb=vectores de coordenadas x y y, respectivamente, de los puntos de la
    %herramienta ordenados de menor a mayor con respecto a los valores del vector pcx
    %Ib=Índices o lugares de los puntos de la herramienta que están en colisión dentro
    %del vector Xb
    %valorHph=El valor de la función phi de los puntos de la herramienta en colisión
    %rig=índice de rigidez
%Salidas:
    %F=Vector final de fuerza aplicado al centro de masas de la herramienta
    %gradx,grady=coordenadas x y y, respectivamente, de los vectores de fuerza
```

```
    %correspondientes a cada uno de los puntos de la herramienta que está en colisión
    %m=Número de puntos en colisión


%----------------------------------------
%Fuerza
F=0;%Inicialización
%Puntos de la herramienta en colisión
m=size(Pcx,2);
%----------------------------------------
%Se barre con todos los puntos que están en colisión
for i=1:m
    %----------------------------------------
    %CÁLCULO DE FUERZAS
    %Se construye los gradientes de cada uno de los vértices de las celdas
    %grad1 corresponde a (Pcx(i),Pcy(i)), grad2 corresponde a (Pcx(i)+1,Pcy(i)),
    %grad3 corresponde a (Pcx(i),Pcy(i)+1) y grad4 corresponde (Pcx(i)+1,Pcy(i)+1).
    if (1<Pcx(i))&&(Pcx(i)<n+1)
        grad1(1)=(ph(Pcx(i)+1,Pcy(i))-ph(Pcx(i)-1,Pcy(i)))/(2*h);
        if (1<=Pcy(i))&&(Pcy(i)<n+1)
            grad3(1)=(ph(Pcx(i)+1,Pcy(i)+1)-ph(Pcx(i)-1,Pcy(i)+1))/(2*h);
            if Pcy(i)~=1
                grad1(2)=(ph(Pcx(i),Pcy(i)+1)-ph(Pcx(i),Pcy(i)-1))/(2*h);
            else
                grad1(2)=(ph(Pcx(i),Pcy(i)+1)-ph(Pcx(i),Pcy(i)))/h;
            end
            if (1<Pcy(i)+1)&&(Pcy(i)+1<n+1)
                grad3(2)=(ph(Pcx(i),Pcy(i)+2)-ph(Pcx(i),Pcy(i)))/(2*h);
            elseif Pcy(i)+1==n+1
                grad3(2)=(ph(Pcx(i),Pcy(i)+1)-ph(Pcx(i),Pcy(i)))/h;
            end
        end
        if (1<Pcx(i)+1)&&(Pcx(i)+1<n+1)
            grad2(1)=(ph(Pcx(i)+2,Pcy(i))-ph(Pcx(i),Pcy(i)))/(2*h);
            if (1<=Pcy(i))&&(Pcy(i)<n+1)
                grad4(1)=(ph(Pcx(i)+2,Pcy(i)+1)-ph(Pcx(i),Pcy(i)+1))/(2*h);
                if Pcy(i)~=1
```

```
                grad2(2)=(ph(Pcx(i)+1,Pcy(i)+1)-ph(Pcx(i)+1,Pcy(i)-1))/(2*h);
            else
                grad2(2)=(ph(Pcx(i)+1,Pcy(i)+1)-ph(Pcx(i)+1,Pcy(i)))/h;
            end
            if (1<Pcy(i)+1)&&(Pcy(i)+1<n+1)
                grad4(2)=(ph(Pcx(i)+1,Pcy(i)+2)-ph(Pcx(i)+1,Pcy(i)))/(2*h);
            elseif Pcy(i)+1==n+1
                grad4(2)=(ph(Pcx(i)+1,Pcy(i)+1)-ph(Pcx(i)+1,Pcy(i)))/h;
            end
        end
    elseif (Pcx(i)+1==n+1)
        grad2(1)=(ph(Pcx(i)+1,Pcy(i))-ph(Pcx(i),Pcy(i)))/h;
        if (1<=Pcy(i))&&(Pcy(i)<n+1)
            grad4(1)=(ph(Pcx(i)+1,Pcy(i)+1)-ph(Pcx(i),Pcy(i)+1))/h;
            if Pcy(i)~=1
                grad2(2)=(ph(Pcx(i)+1,Pcy(i)+1)-ph(Pcx(i)+1,Pcy(i)-1))/(2*h);
            else
                grad2(2)=(ph(Pcx(i)+1,Pcy(i)+1)-ph(Pcx(i)+1,Pcy(i)))/h;
            end
            if (1<Pcy(i)+1)&&(Pcy(i)+1<n+1)
                grad4(2)=(ph(Pcx(i)+1,Pcy(i)+2)-ph(Pcx(i)+1,Pcy(i)))/(2*h);
            elseif Pcy(i)+1==n+1
                grad4(2)=(ph(Pcx(i)+1,Pcy(i)+1)-ph(Pcx(i)+1,Pcy(i)))/h;
            end
        end
    end
elseif (Pcx(i)==1)
    grad1(1)=(ph(Pcx(i)+1,Pcy(i))-ph(Pcx(i),Pcy(i)))/h;
    if (1<=Pcy(i))&&(Pcy(i)<n+1)
        grad3(1)=(ph(Pcx(i)+1,Pcy(i)+1)-ph(Pcx(i),Pcy(i)+1))/h;
        if Pcy(i)~=1
            grad1(2)=(ph(Pcx(i),Pcy(i)+1)-ph(Pcx(i),Pcy(i)-1))/(2*h);
        else
            grad1(2)=(ph(Pcx(i),Pcy(i)+1)-ph(Pcx(i),Pcy(i)))/h;
        end
        if (1<Pcy(i)+1)&&(Pcy(i)+1<n+1)
```

```
                grad3(2)=(ph(Pcx(i),Pcy(i)+2)-ph(Pcx(i),Pcy(i)))/(2*h);
            elseif Pcy(i)+1==n+1
                grad3(2)=(ph(Pcx(i),Pcy(i)+1)-ph(Pcx(i),Pcy(i)))/h;
            end
        end
        if (1<Pcx(i)+1)&&(Pcx(i)+1<n+1)
            grad2(1)=(ph(Pcx(i)+2,Pcy(i))-ph(Pcx(i),Pcy(i)))/(2*h);
            if (1<=Pcy(i))&&(Pcy(i)<n+1)
                grad4(1)=(ph(Pcx(i)+2,Pcy(i)+1)-ph(Pcx(i),Pcy(i)+1))/(2*h);
                if Pcy(i)~=1
                    grad2(2)=(ph(Pcx(i)+1,Pcy(i)+1)-ph(Pcx(i)+1,Pcy(i)-1))/(2*h);
                else
                    grad2(2)=(ph(Pcx(i)+1,Pcy(i)+1)-ph(Pcx(i)+1,Pcy(i)))/h;
                end
                if (1<Pcy(i)+1)&&(Pcy(i)+1<n+1)
                    grad4(2)=(ph(Pcx(i)+1,Pcy(i)+2)-ph(Pcx(i)+1,Pcy(i)))/(2*h);
                elseif Pcy(i)+1==n+1
                    grad4(2)=(ph(Pcx(i),Pcy(i)+1)-ph(Pcx(i)+1,Pcy(i)))/(2*h);
                end
            end
        end
    end
    %----------------------------------------
    %CÁLCULO DE GRADIENTES Y NORMALES POR INTERPOLACIÓN
    %Se encuentra el gradiente de la función ph en los puntos de la herramienta que
    %están en colisión
    grad=(grad1*(X(Pcx(i)+1)-Xb(Ib(i)))*(Y(Pcy(i)+1)-Yb(Ib(i)))+
grad2*(Xb(Ib(i))-X(Pcx(i)))*(Y(Pcy(i)+1)-Yb(Ib(i)))+
grad3*(X(Pcx(i)+1)-Xb(Ib(i)))*(Yb(Ib(i))-Y(Pcy(i)))+
grad4*(Xb(Ib(i))-X(Pcx(i)))*(Yb(Ib(i))-Y(Pcy(i))))
/((X(Pcx(i)+1)-X(Pcx(i)))*(Y(Pcy(i)+1)-Y(Pcy(i))));
    %Se extrae la dirección de este vector (vector unitario) y de le agrega el módulo
    %igual valor absoluto del nÃºmero de ph, valorHph(i) calculado previamente
    %En caso de que se anule el valor anterior se pued escoger su valor aleatoriamente
    %como se hace rrecurriendo a los gradientes más cercanos como se hace en las
    %últimas cuatro opciones
```

```
if norm(grad)~=0
    %Dirección
    gradx(i)=grad(1)/norm(grad);
    grady(i)=grad(2)/norm(grad);
    grad=grad/norm(grad);
    %Módulo
    gradx(i)=abs(valorHph(i))*gradx(i);
    grady(i)=abs(valorHph(i))*grady(i);
    grad=abs(valorHph(i))*grad;
elseif norm(grad1)~=0
    %Dirección
    gradx(i)=grad1(1)/norm(grad1);
    grady(i)=grad1(2)/norm(grad1);
    %Módulo
    gradx(i)=abs(valorHph(i))*gradx(i);
    grady(i)=abs(valorHph(i))*grady(i);
    grad=abs(valorHph(i))*grad1;
elseif norm(grad2)~=0
    %Dirección
    gradx(i)=grad2(1)/norm(grad2);
    grady(i)=grad2(2)/norm(grad2);
    %Módulo
    gradx(i)=abs(valorHph(i))*gradx(i);
    grady(i)=abs(valorHph(i))*grady(i);
    grad=abs(valorHph(i))*grad2;
elseif norm(grad3)~=0
    %Dirección
    gradx(i)=grad3(1)/norm(grad3);
    grady(i)=grad3(2)/norm(grad3);
    %Módulo
    gradx(i)=abs(valorHph(i))*gradx(i);
    grady(i)=abs(valorHph(i))*grady(i);
    grad=abs(valorHph(i))*grad3;
elseif norm(grad4)~=0
    %Dirección
    gradx(i)=grad4(1)/norm(grad4);
```

```
        grady(i)=grad4(2)/norm(grad4);
        %Módulo
        gradx(i)=abs(valorHph(i))*gradx(i);
        grady(i)=abs(valorHph(i))*grady(i);
        grad=abs(valorHph(i))*grad4;
    end
    %---------------------------------------
    %SUMA DE FUERZAS RESULTANTES
    F=F+rig*grad;%Se agrega la rigidez
end
%---------------------------------------
%PROMEDIO DE FUERZAS RESULTANTES
F=F/m;
```

A.5. Funciones auxiliares

```
%NORMA DEL GRADIENTE
function [gr]=normgrad(ph,n,h)
%Con esta función se calcula la norma del gradiente de una función implícita ph en una
%malla de n+1xn+1
%Entradas:
    %ph=La función ph
    %n=Número de puntos en que se discretiza los ejes coordenados
    %h=Tamaño de paso espacial
%Salidas:
    %gr=El gradiente de phi

%CÁLCULO DE LA NORMA DEL GRADIENTE
%El gradiente de los puntos que no están en la frontera se calcula con diferencias
%centradas
gr(2:n,2:n)=sqrt((ph(3:n+1,2:n)-ph(1:n-1,2:n)).^2/(2*h)^2+
(ph(2:n,3:n+1)-ph(2:n,1:n-1)).^2/(2*h)^2);
%El gradiente de la frontera se calcula a partir de condiciones de frontera adiabáticas
gr(1,:)=0;
gr(n+1,:)=0;
gr(:,1)=0;
```

```
gr(:,n+1)=0;

%FUNCIÓN DELTA CONTINUA
function [delta]=dlcon(ph,n,h)
%Con esta función se construye en una malla n+1xn+1 una aproximación continua a la
%función delta de una función implícita ph
%Entradas:
    %ph=La función ph
    %n=Número de puntos en que se discretiza los ejes coordenados
    %h=Tamaño de paso espacial
%Salidas:
    %delta=Función delta

%Parámetro que determina la longitud de banda de aproximación a la función delta
epsilon=1.5*h;
%----------------------------------------
%CONSTRUCCIÓN DE LA FUNCIÓN DELTA
for i=1:n+1
  for j=1:n+1
      if (-epsilon<=ph(i,j))&&(ph(i,j)<=epsilon)
          delta(i,j)=1/(2*epsilon)+(1/(2*epsilon))*cos(pi*ph(i,j)/epsilon);
      else
          delta(i,j)=0;
      end
   end
end

%VOLUMEN FUERA DE LA HERRAMIENTA
function [Vol]=Volg(ph,ph2,n,h)
%Esta función calcula el "volumen" del material deformable que se encuentra fuera de
%la herramienta
%Entradas:
    %ph=La función ph
    %ph2=La malla que define el exterior de la herramienta con valor '1' y '0' donde se
    %encuentra esta
    %n=Número de puntos en que se discretiza los ejes coordenados
```

```
    %h=Tamaño de paso espacial
%Salidas:
    %Vol=Volumen del material deformable fuera de la herramienta

%----------------------------------------
%Volumen
Vol=0;%Inicialización
%----------------------------------------
%CONSTRUCCIÓN DEL VOLUMEN
for i=1:n+1
  for j=1:n+1
      if (ph(i,j)<=0)&&(ph2(i,j)==1)
          Vol=Vol+1;
      end
  end
end
Vol=Vol*h*h;%Volumen

%ORDENACIÓN DE LOS PUNTOS DE LA HERRAMIENTA px
function [Xb,Yb]=orden(px,py,k);
%Función que se encarga de ordenar los puntos de la herramienta de menor a mayor con
%respecto al vector de coordenadas px
%Entradas:
    %px,py=son los vectores de coordenadas x y y, respectivamente, de los puntos de la
    %herramienta
    %k=Número de puntos de la herramienta
%Salidas:
    %Xb,Yb=vectores de coordenadas x y y, respectivamente, de los puntos de la
    %herramienta ordenados de menor a mayor con respecto a los valores del vector pcx

%----------------------------------------
%ORDENAMIENTO
for i=1:k
    Xb(i)=px(i);
    Yb(i)=py(i);
    for j=1:i-1
```

```
        if Xb(i-j)<=Xb(i-j+1)
            break;
        elseif Xb(i-j)>Xb(i-j+1)
            temp=Xb(i-j);
            Xb(i-j)=Xb(i-j+1);
            Xb(i-j+1)=temp;
            temp=Yb(i-j);
            Yb(i-j)=Yb(i-j+1);
            Yb(i-j+1)=temp;
        end
    end
end
```

Bibliografía

[1] M. C. Lin y W. V. Baxter, "Modeling and Creative Processes", en Haptic Rendering: Foundations, Algorithms and Aplications, pp. 531-548, A. K. Peters, Ltd., 2008.

[2] R. L. Klatzky y S. J. Lederman, "Handbook of Psychology: Experimental Psychology", John Wiley & Sons, New York, 2003.

[3] T. A. Kern. "Terminology", en Engeenering Haptic Devices, pp. 19-33, Springer, 2009.

[4] K. Salisbury, F. Conti y F. Barbagly, "Haptic Rendering:Introductory Concepts", en IEEE Computer Graphics and Applications, pp. 24-32, Marzo/Abril 2004.

[5] K. E. Maclean y V. Hayward, "Do It Yourself Haptics: Part II. Interaction Design ", en IEEE Robotics & Automation Magazine, pp. 104-118, Marzo 2008.

[6] M. A. Otaduy y M. C. Lin, "Introduction to Haptic Rendering Algorithms", en Haptic Rendering: Foundations, Algorithms and Aplications, pp. 159-180, A. K. Peters, Ltd., 2008.

[7] S. Redon, "Continuous Collision Detection", en Haptic Rendering: Foundations, Algorithms and Aplications, pp. 253-276, A. K. Peters, Ltd., 2008.

[8] C. Duriez, "Rendering of Frictional Contact with Deformable Enviroments", en Haptic Rendering: Foundations, Algorithms and Aplications, pp. 421-441, A. K. Peters, Ltd., 2008.

[9] M. A. Weiss, "Estructuras de Datos en Java", Addisson Wesley, 2000.

[10] R. C. McOwen, "Partial Diferential Equations", Prentice Hall, 2003.

[11] J. D. Foley, A. v. Dam, S. K. Feiner, J. F. Hughes y R. L. Phillips, "Introduction to Computer Graphics", Addison Wesley, 1994.

[12] C. Prieto, "Topología básica", Fondo de cultura económica, 2003.

[13] R. L. Wilder, "Topology of Manifolds", American Mathematical Society Colloquium Publications 32, 1949.

[14] Z. Gao e I. Gibson, "Haptic B-spline Surface Sculpting with Shaped Tool of Implicit Surface", en Computer-Aided Design & Applications, pp 263-272, Vol. 2, Nos. 1-4, 2005.

[15] F. Dachille, H. Qin y A. Kaufman, "A novel haptics-based interface and sculpting system for physics-based geometric design", en Computer-Aided Design, vol. 33, pp. 403-420, 2001.

[16] C. H. Ho, C. Basdogam y M. A. Srinivasan, "Haptic rendering: point- and ray-based interactions", presented at the second PHANToM Users Group Workshop, 1997.

[17] L. Kim, A. Kyrikou, G. S. Sukhatme y M. Desbrun, "An Implicit-based Haptic Rendering Technique", presented at IEEE IROS 2002, Switzerland, 2002.

[18] R. Raffin, G. Gesquière, E. Remy y S. Thon, "VirSculpt: a virtual sculpting environment", en GraphiCon' 04 Proceedings, 2004, 184-187.

[19] A. Bærentzen, "Octree-based volume sculpting", presentado en IEEE Visualization '98, Late Breaking Hot Topics Proceedings, IEEE Computer Society Press, pp. 9-12, 1998.

[20] E. Ferley, M.-P. Cani y J.-D. Gascuel, "Practical Volumetric Sculpting", en The Visual Computer, Volume 16(8), pp. 469–480, december 2000. Una versión preliminar de este documento apareció en Implicit Surfaces'99, Bordeaux, Franccia, Septiembre 1999.

[21] S. Mizuno, M. Okada y J. Toriwaki, "Virtual sculpting and virtual woodcut printing", en The Visual Computer, Volume 14(2),

[22] Y.-H. Chai, G.R. Luecke y J. C. Edwards, "Virtual clay modeling using the isu exoskeleton", en proceedings of VRAIS'98, pp. 39-51, Marzo 1998.

[23] J.-A. Thingvold y E. Cohen, "Physical modeling with B-splines surfaces for interactive design and animation", en Computer Graphics, 24(4):129–137, en Proceedings of SIGGRAPH'90 (Dallas), Agosto 1990.

[24] T.I. Vassilev, "Interactive sculpting with deformable non uniform b-splines", en Computer Graphics Forum, 16(4):191–199, 1997.

[25] R. S. Avila y L.M. Sobierajski, "A haptic interaction method for volume visualization ", en Computer Graphics, pp. 197–204, Octubre 1996. Proceedings of Visualization'96 (San Francisco).

[26] T. A. Galyean y J. F. Hughes, "Sculpting: An Interactive Volumetric Modeling Technique", en Computer Graphics, Volume 25, Number 4, pp. 268-274, Julio 1991.

[27] T. Massie, "A tangible goal for 3d modeling", en IEEE Computer Graphics and Applications, 3:62–65, Mayo 1998.

[28] E. Ferley, M. P. Cani y J.-D. Gascuel, "Virtual Sculpture", Short Papers Proceedings of Eurographics'99, Milán, Italia, 1999.

[29] J.-R. Bill y S.K. Lodha, "Sculpting polygonal models using virtual tools", en Graphics Interface'95, pp. 272–278, 1995.

[30] K. T. McDonell y Hong Qin, "Virtual Clay: Haptics-based Deformable Solids of Arbitrary Topology"

[31] G. Dewaele y M.-P. Cani, "Interactive Global and Local Deformations for virtual Clay", en 11 th Pacific Conference on Computer Graphics and Applications (Pacific Grafics), pp. 131-140, 2003.

[32] S. Druon, A.Crosnier y L. Brigandat, "Eficient cellular automata for 2d / 3d free-form modelin", WSCG, 11, Febrero 2003.

[33] H. Arata, Y. Takai, N. K. Takai y T. Yamamoto, "Free-form shape modeling by 3d cellular automata", en International Conference on Shape Modeling and Applications, pp. 242–247, 1999.

[34] E. Grinspun, P. Krysl y P. Schröder, "Charms: A simple framework for adaptive simulatio", en ACM Transactions on Graphics, 21(3):281–290, Julio 2002. Proceedings of ACM SIGGRAPH 2002.

[35] K. T. McDonnell, H. Qin and R. A. Wlodarczyk, "Virtual clay: a real-time sculpting system with haptic toolkits", en ACM Symposium on Interactive 3D Graphics, pp. 179–190, Marzo 2001.

[36] J. A. Sethian, "Theory, Algorithms, and Applications of Level Set Methods for Propagating Interfaces", 1995.

[37] Y. Duan, J. Hua y H. Qin, "HapticFlow: PDE-based mesh editing with haptics", en Comp. Anim. Virtual Worlds 2004, pp. 193-200, 2004.

[38] J. A. Sethian, "Level Set Techniques for Tracking Interfaces; Fast Algorithms, Multiple Regions, Grid Generation, and Shape/Character Recognition", 2010.

[39] S. Osher y J. A. Sethian, "Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulation", en Jour. Comp. Phys, pp. 12-49, 1988.

[40] J. A. Sethianm, "Level Set Methods and Fast Marching Methods", Cambridge University Press, 1999.

[41] R. Blanch, E. Ferley, M.-P. Cani y J.-D. Gascuel, "Non-realistic haptic feedback for virtual sculpture", Technical Report RR-5090, INRIA, U.R. Rhone-Alpes, Enero 2004.

[42] R. Jagnou y J. Dorsey, "Virtual Sculpting with Haptic Displacement Maps", 2002.

[43] J. A. Bæretzen y N. J. Christensen, "Volume Sulptin Using the Level-Set Method", en Proceedings of the Shape Modeling International 2002 (SMI'02), p. 175, 2002.

[44] X. Guo, J. Hua y H. Qin, "Point Set Surfaces Techniques based on Level-Sets", en Proc. CGI, 2004.

[45] A. Gregroy, S. Ehmann y M. C. Lin, "InTouch: Interactive Multiresolution Modeling and 3D Painting with a Haptic Interface", en the Proceedings of IEEE Virtual Reality Conference 2000, 2000.

[46] J. Munkress, "Topology", Prentice Hall, 2000.

[47] J. M. Lee, "Introduction to topological Manifolds", Springer, 2000.

[48] M. C. Delfour y J.-P. Zolésio, "Shapes and Geometries: Metrics, Analysis, Differential Calculus and Otimization", Advances in Design and Control, siam, 2011.

[49] S. Osher y R. Fedkiw, "Level Set Mehtods and Dynamic Implicit Surfaces", Springer, 2002.

[50] C. W. Shu y S. Osher, "Efficient Implementation of Essentially Non-Oscillatory Shock Capturing Schemes", en J. Comput. Phys. 77, pp. 439-471, 1988.

[51] D. Chopp, "Computing Minimal Surfaces via Level Set Curvature Flow", en J. Comput. Phys. 106, pp. 77-91, 1993.

[52] J. Tsitsiklis, "Efficient Algorithms for Globally Optimal Trajectories", en IEEE Transactions on Automatic Control 40, pp. 1528-1538, 1995.

[53] L. Leithold, "El cálculo", Oxford University Press, 1998.

[54] M. Giaquinta y S. Hildebrandt, "Calculus of Variations I", segunda edición, Springer, 2004.

[55] R. Strichartz, "A Guide to Distribution Theory and Fourier Transforms", CRC Press, 1994.

[56] J., Strikwerda, "Finite Difference Schemes and Partial Differential Equations", SIAM, 2004.

[57] M. Sussman, P. Smereka, S. Osher, "A Lecel Set Approach for Computing Solutions to Incompressible Two-Phase Flow", J. Comput Phys 114, 146-159 (1994).