Universidad Politécnica de Cartagena

industriales etsii UPCT

Simulación numérica de inclusiones subdiferenciales

Proyecto Fin de Carrera

E.T.S. Ingeniería Industrial

Luis Francisco Ruiz Ortín

Ingeniero Industrial

Intensificaciones: Mecánica y Fabricación, Construcciones e Instalaciones Industriales

Julio 2014

José Alberto Murillo Hernández (Director)

Agradecimientos

A mi familia por el apoyo recibido estos largos años de esfuerzos.

A Alberto por la inmensa ayuda prestada y la infinita paciencia mostrada durante la realización de este proyecto.

A Claudia, por acompañarme siempre.

Indice

Introducción

Este Proyecto Fin de Carrera surge gracias a la concesión por parte del Ministerio de Educación, Cultura y Deporte de una beca de colaboración en el Departamento de Matemática Aplicada y Estadística de la Universidad Politécnica de Cartagena bajo la dirección del profesor Dr. José Alberto Murillo Hernández.

Dicha beca estaba enfocada a profundizar en el análisis numérico de las inclusiones subdiferenciales o ecuaciones diferenciales multivaluadas de tipo subdiferencial. Estos objetos surgen a partir de las ecuaciones diferenciales de tipo gradiente

$$-\dot{\mathbf{x}}(t) = \nabla\phi(\mathbf{x}(t))$$

que son bien conocidas por matemáticos e ingenieros, dado que

- Permiten obtener trayectorias de descenso del potencial ϕ
- Describen sistemas dinámicos de interés
- Aparecen en la modelización del comportamiento de cierto tipo de materiales

Sin embargo, tanto desde el punto de vista teórico como de las aplicaciones, surgen problemas en los que el potencial ϕ no es una función regular (derivable). No obstante, cuando es convexa puede definirse el concepto de subgradiente de ϕ como una generalización del gradiente y plantear ecuaciones del tipo anterior. El problema es que el subgradiente en cada punto no es necesariamente único, en realidad esto sólo ocurre cuando el potencial se puede derivar, lo que obliga a definir la subdiferencial como el conjunto de los subgradientes y a considerar expresiones del tipo

$$-\dot{\mathbf{x}}(t) \in \partial\phi(\mathbf{x}(t))$$

que es lo que se conoce como inclusiones subdiferenciales.

El objeto de este proyecto es avanzar en el estudio de la aproximación numérica de ecuaciones diferenciales con términos multivaluados generados por subdiferenciales de funciones (potenciales) convexas. Cabe decir que este es un tema poco tratado y, paradójicamente, a pesar de la abundante bibliografía sobre los aspectos teóricos de las inclusiones diferenciales y sobre su relación con modelos matemáticos de fenómenos físicos, no existen muchas referencias en las que se estudie su tratamiento numérico de forma general.

Cabe mencionar aquí la monografía [13] en la que se aborda una primer análisis numérico del problema que nos ocupa y que constituye el punto de partida de la presente memoria.

La idea básica que desarrollamos en el proyecto, es usar la regularización de Yosida para obtener un "problema regularizado" que en cierta forma constituye una aproximación del problema original y a continuación utilizar métodos numéricos de aproximación de ecuaciones diferenciales ordinarias para resolver el nuevo problema regularizado.

La idea de utilizar la regularización de Yosida como aproximación de la subdiferencial es clásica en el campo del Análisis Convexo y se ha venido utilizando desde los años 70 del siglo pasado para obtener demostraciones teóricas de existencia de solución. Sin embargo, sorprendentemente, no se han explorado convenientemente sus implicaciones numéricas.

En realidad la aproximación numérica de las soluciones de este tipo de inclusiones diferenciales se ha basado fundamentalmente en dos ideas

- Métodos *ad hoc* para cada caso concreto que aproximan la función convexa mediante una función regular que les permite resolver el problema aproximado, pero que no proporcionan una etodología general que pueda ser usada de forma sistemática. Además suelen usarse en casos relativamente sencillos en que la función convexa solamente depende de una variable (por ejemplo, el valor absoluto).

- Obtener selecciones de la subdiferencial y aplicar métodos de discretización, o que requiere conocer, aunque sea de forma aproximada, el conjunto subdiferencial en cada punto, lo que no es en absoluto elemental, por ejemplo cuando se tienen funciones convexas definidas mediante supremos de familias de funciones.

Frente a estas técnicas, la metodología que proponemos en este proyecto es válida para cualquier función potencial convexa, independientemente de su número de variables y no es necesario conocer en absoluto la estructura del conjunto subdiferencial.

Se trata además de una técnica general, ya que se puede aplicar a múltiples problemas y flexible ya que el cálculo de la regularización de Yosida puede combinarse con cualquier método numérico de aproximación de ecuaciones diferenciales ordinarias.

Esta memoria comienza desarrollando los fundamentos teóricos acerca del análisis convexo e inclusiones diferenciales asociadas a subdiferenciales. Hemos decidido incluir un resumen bastante amplio de los principales conceptos y resultados dado que se trata de un tópico que no está incluido en los contenidos de Matemáticas de los estudios de Ingeniería.

Una vez comentados los conceptos teóricos para la comprensión de la memoria, en el siguiente capítulo denominado Algoritmos y códigos se ha descrito la

estructura de los algoritmos empleados, donde tras utilizar la regularización de Yosida para aproximar la subdiferencial, se han utilizando los métodos numéricos de Euler y Runge-Kutta. También se introducen y comentan los códigos programados en MATLAB de los mencionados algoritmos. Se describen así mismo diferentes escenarios según el tipo de problema y se desarrollan simulaciones numéricas que permiten validar el correcto funcionamiento de los códigos.

El Capítulo 3 se dedica a la presentación de la interfaz gráfica desarrollada en la que están implementados todos los casos estudiados y permite la introducción de manera sencilla y sin conocimientos profundos de programación de todos los parámetros necesarios para la resolución de diferentes tipo de problemas asociados a subdiferenciales.

A continuación se aborda el estudio detallado de los problemas asociados a un caso concreto de potenciales, problemas que denominamos genéricamente de persecución. Este tipo de problemas presentan desafíos tanto desde el punto de vista teórico (entender el comportamiento de las soluciones), como de la modelización (esta clase de problemas están asociados a la evolución de sistemas en los que se desea que el estado verifique ciertas restricciones) y también computacionales (ha sido necesario, por ejemplo, modificar los códigos para poder tratar esta clase particular de problemas). En este capítulo se incluyen tanto los algoritmos utilizados para cada método numérico como la interfaz creada expresamente para este caso concreto y al igual que cuando se hace el estudio general, se incluyen ejemplos que permiten validar el funcionamiento de los códigos.

Por último se incluyen en un Anexo todos los códigos de los programas creados en MATLAB.

Capítulo 1

Fundamentos teóricos

En este capítulo, como su nombre indica, se recopilan las nociones teóricas que se manejarán a lo largo de la memoria, tanto las definiciones y propiedades básicas, como diversos resultados que se establecen a partir de las mismas. El capítulo consta de cuatro apartados que se agrupan en dos bloques: los tres primeros pueden englobarse bajo el epígrafe de *Análisis Convexo*, mientras que en el último apartado se presentan los diversos tipos de ecuaciones diferenciales multivaluadas o inclusiones diferenciales asociadas a subdiferenciales que se consideran en la memoria.

La referencia básica en el campo del Análisis Convexo es el libro de Rockafellar [15]. Son también remarcables los textos de Hiriart-Urruty y Lemaréchal [9], que tiene un estilo más accesible, y Aubin [2], que constituye una buena introducción al tema. Una tratamiento más avanzado puede encontrarse en [5] y [16].

En cuanto a las inclusiones diferenciales, el texto de referencia es el de Aubin y Cellina [3], que dedica el tercer capítulo a las inclusiones de tipo subdiferencial con potencial constante a lo largo del tiempo, una materia introducida por Brézis en el clásico [4], aunque sea un texto difícil por su enfoque abstracto. El caso de las inclusiones subdiferenciales con potenciales dependiendo del tiempo y los problemas bipotenciales se estudia con detalle en el capítulo II del manual enciclopédico de Hu y Papageorgiou [10].

Finalmente, cabe decir que en [9], [10], [15] y [16] pueden encontrarse múltiples comentarios y referencias sobre el desarrollo histórico de los contenidos del presente capítulo.

1.1 Conjuntos convexos

El estudio sistemático de los conjuntos convexos y sus propiedades fue iniciado a principios del siglo pasado por Minkowski, motivado fundamentalmente

por cuestiones geométricas, aunque una cierta noción de convexidad aparece en trabajos clásicos de disciplinas de carácter *aplicado* como la mecánica y la termodinámica.

1.1.1 Algunas nociones básicas

Sobre el espacio vectorial $\mathbb{R}^N$ se considera el producto escalar canónico, denotado por

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{j=1}^{N} x_j y_j \tag{1.1}$$

para cada $\mathbf{x} = (x_1, \ldots, x_N)$, $\mathbf{y} = (y_1, \ldots, y_N)$ en $\mathbb{R}^N$ y la norma asociada (conocida como *norma euclídea*)

$$|\mathbf{x}| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}, \quad \mathbf{x} \in \mathbb{R}^N \tag{1.2}$$

que permite definir la distancia entre dos puntos como la norma de su diferencia $d(\mathbf{x}, \mathbf{y}) = |\mathbf{y} - \mathbf{x}|$. También se definen una clase especial de conjuntos, denominados *bolas*, mediante la relación

$$B_\delta(\mathbf{x}) = \left\{ \mathbf{y} \in \mathbb{R}^N : |\mathbf{y} - \mathbf{x}| < \delta \right\}. \tag{1.3}$$

En particular, el conjunto anterior es la *bola abierta de centro* $\mathbf{x}$ *y radio* $\delta > 0$. Cuando en lugar de una desigualdad estricta se tiene el símbolo $\leq$ (es decir, se incluyen en el conjunto los puntos que distan del centro exactamente el radio) se habla de *bola cerrada*, que denotaremos $\overline{B}_\delta(\mathbf{x})$. Cuando $\mathbf{x} = \mathbf{0}$, $\delta = 1$, se habla de la *bola unidad*, denotada por $\mathcal{B} = \overline{B}_1(\mathbf{0})$.

Dado un conjunto no vacío, $C \subset \mathbb{R}^N$, se define la distancia de un punto arbitrario $\mathbf{x} \in \mathbb{R}^N$ a C como

$$d_C(\mathbf{x}) = \inf \left\{ |\mathbf{x} - \mathbf{z}| : \mathbf{z} \in C \right\}. \tag{1.4}$$

Obviamente $0 \leq d_C(\mathbf{x}) < +\infty$, con $d_C(\mathbf{x}) = 0$ si y solamente si $\mathbf{x}$ pertenece a la clausura, $\overline{C}$, del conjunto C. De la definición de la función distancia es evidente que para cada $\mathbf{x} \in \mathbb{R}^N$ existirá una sucesión minimizante $\{\mathbf{z}_m\}$ en C de forma que $|\mathbf{z}_m - \mathbf{x}| \to d_C(\mathbf{x})$. Así, dado $\mathbf{y} \in \mathbb{R}^N$ se tiene que

$$d_C(\mathbf{y}) \leq |\mathbf{z}_m - \mathbf{y}| \leq |\mathbf{z}_m - \mathbf{x}| + |\mathbf{x} - \mathbf{y}|$$

de donde tomando límite cuando $m \to \infty$ se llega a la desigualdad

$$d_C(\mathbf{y}) \leq d_C(\mathbf{x}) + |\mathbf{y} - \mathbf{x}|. \tag{1.5}$$

Obviamente también se verifica la desigualdad recíproca, por lo que podemos escribir

$$|d_C(\mathbf{x}) - d_C(\mathbf{y})| \leq |\mathbf{x} - \mathbf{y}| \tag{1.6}$$

es decir, la función distancia a un conjunto es Lipschitz.

Dados dos conjuntos $A, C \subset \mathbb{R}^N$, se define su *distancia* en el sentido de Hausdorff mediante la relación

$$\mathbf{dl}_{\mathcal{H}}(A, C) := \max\left(\sup_{\mathbf{x}\in A} d_C(\mathbf{x}), \sup_{\mathbf{z}\in C} d_A(\mathbf{z})\right). \tag{1.7}$$

Obviamente $0 \leq \mathbf{dl}_{\mathcal{H}}(A, C) \leq +\infty$, con $\mathbf{dl}_{\mathcal{H}}(A, C) = 0$ si y solamente si $\overline{A} = \overline{C}$.

En el caso de conjuntos acotados, la distancia de Hausdorff es siempre finita. Si consideramos la familia $\mathcal{K}(\mathbb{R}^N)$ de los subconjuntos compactos (cerrados y acotados) de $\mathbb{R}^N$, se prueba que $\mathbf{dl}_{\mathcal{H}}$ verifica las propiedades usuales de una distancia:

1. $0 \leq \mathbf{dl}_{\mathcal{H}}(K, Q) < +\infty$, para cada $K, Q \in \mathcal{K}(\mathbb{R}^N)$, con $\mathbf{dl}_{\mathcal{H}}(K, Q) = 0$ si y solamente si $K = Q$.
2. $\mathbf{dl}_{\mathcal{H}}(K, Q) = \mathbf{dl}_{\mathcal{H}}(Q, K)$, para cada $K, Q \in \mathcal{K}(\mathbb{R}^N)$.
3. Dados $K, Q, C \in \mathcal{K}(\mathbb{R}^N)$, se tiene $\mathbf{dl}_{\mathcal{H}}(K, Q) \leq \mathbf{dl}_{\mathcal{H}}(K, C) + \mathbf{dl}_{\mathcal{H}}(Q, C)$.

Si consideramos la suma de dos conjuntos, definida como,

$$A + C = \{\mathbf{x} + \mathbf{z} : \mathbf{x} \in A,\ \mathbf{z} \in C\} \tag{1.8}$$

y, si $0 < \mathbf{dl}_{\mathcal{H}}(A, C) < +\infty$, de la definición de distancia de Hausdorff se deducen las inclusiones

$$A \subseteq C + \mathbf{dl}_{\mathcal{H}}(A, C)\mathcal{B}, \qquad C \subseteq A + \mathbf{dl}_{\mathcal{H}}(A, C)\mathcal{B}.$$

Finalmente, para cada $\mathbf{x} \in \mathbb{R}^N$ y cada par de conjuntos $A, C \subset \mathbb{R}^N$, se verifica la desigualdad

$$|d_A(\mathbf{x}) - d_C(\mathbf{x})| \leq \mathbf{dl}_{\mathcal{H}}(A, C) \tag{1.9}$$

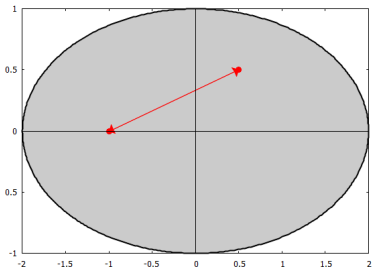
es decir, la aplicación $A \rightsquigarrow d_A(\mathbf{x})$ es Lipschitz.

1.1.2 Conjuntos convexos

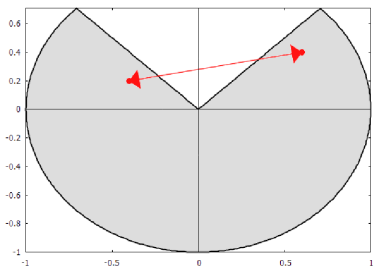
Dados dos puntos $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$, se llama combinación convexa a cualquier otro punto de la forma

$$(1 - \lambda)\mathbf{x} + \lambda\mathbf{y}$$

donde $0 \leq \lambda \leq 1$. Geométricamente estos puntos corresponden al segmento rectilíneo que une los puntos $\mathbf{x}$ e $\mathbf{y}$. Un conjunto $D \subset \mathbb{R}^N$ se dice que es *convexo* si dados dos puntos en D cualquier combinación convexa de los mismos sigue perteneciendo al conjunto, es decir, si $\mathbf{x}, \mathbf{y} \in D$, se tiene que, para cada $0 \leq \lambda \leq 1$, $(1 - \lambda)\mathbf{x} + \lambda\mathbf{y} \in D$. El significado geométrico de esta definición es que un conjunto convexo contiene todos los segmentos uniendo un par de puntos arbitrarios (ver Figura 1.1).



Conjunto convexo



Conjunto no convexo

Figura 1.1:

Ejemplo 1.1 Veamos que la bola unidad $\mathcal{B}$ en $\mathbb{R}^N$ es un conjunto convexo. Tomemos para ello dos puntos $\mathbf{x}, \mathbf{y} \in \mathcal{B}$ y sea $0 \leq \lambda \leq 1$. Obviamente

$$|(1-\lambda)\mathbf{x} + \lambda\mathbf{y}| \leq (1-\lambda)|\mathbf{x}| + \lambda|\mathbf{y}| \leq 1$$

luego $(1-\lambda)\mathbf{x} + \lambda\mathbf{y} \in \mathcal{B}$.

Ejemplo 1.2 Dados $\mathbf{n}_j \in \mathbb{R}^N$, $\alpha_j \in \mathbb{R}$, $j \in I$, con I un subconjunto arbitrario de índices, se define el conjunto

$$D = \left\{\mathbf{x} \in \mathbb{R}^N : \langle \mathbf{n}_j, \mathbf{x}\rangle \leq \alpha_j,\ j \in I\right\}$$

Veamos que es convexo, para lo que se toman dos puntos arbitrarios $\mathbf{x}, \mathbf{y} \in D$, un escalar $0 \leq \lambda \leq 1$ y para cada $j \in I$ se evalúa el producto escalar

$$\langle \mathbf{n}_j, (1-\lambda)\mathbf{x} + \lambda\mathbf{y}\rangle = (1-\lambda)\langle \mathbf{n}_j, \mathbf{x}\rangle + \lambda\langle \mathbf{n}_j, \mathbf{y}\rangle \leq (1-\lambda)\alpha_j + \lambda\alpha_j = \alpha_j$$

que nos indica que $(1-\lambda)\mathbf{x} + \lambda\mathbf{y} \in D$ y, por tanto, el conjunto es convexo.

Cuando el conjunto de índices es finito, es decir, $I = \{1, \ldots, m\} \subset \mathbb{N}$, el conjunto D se denomina *poliedro* o *conjunto poliédrico*. Cuando solamente se tiene una restricción de habla de un *semiplano*.

Los conjuntos convexos tienen importantes propiedades, entre ellas:

- Dada una familia arbitraria $\{A_i\}_{i\in I}$ de conjuntos convexos, su intersección es convexa.
- La imagen por una transformación afín de un conjunto convexo es un convexo, es decir, si $C \subset \mathbb{R}^N$ es convexo, $M \in \mathcal{M}_{N'\times N}(\mathbb{R})$ es una matriz y $\mathbf{z} \in \mathbb{R}^{N'}$, se tiene que

  $$\mathbf{z} + MC = \{\mathbf{z} + M\mathbf{x} : \mathbf{x} \in C\} \subset \mathbb{R}^{N'}$$

  es un conjunto convexo.
- Si C es convexo, su interior y clausura también lo son.

1.1.3 Envolturas convexas

Dado un conjunto arbitrario $C \subset \mathbb{R}^N$, se define su *envoltura convexa*, $\mathrm{co}(C)$, como el menor conjunto convexo que lo contiene. También suele considerarse la *envoltura convexa cerrada*, $\overline{\mathrm{co}}(C)$ que es el menor conjunto convexo y cerrado conteniendo a C.

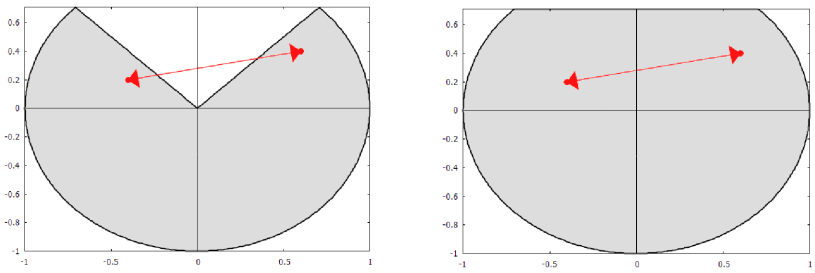
Ejemplo 1.3 Si consideramos el conjunto de la derecha en la Figura 1.1, que puede describirse como

$$\mathcal{O} = \left\{ (r\cos(\theta), r\,\mathrm{sen}(\theta)) \in \mathbb{R}^2 : 0 \le r \le 1,\ \frac{3\pi}{4} \le \theta \le \frac{9\pi}{4} \right\}$$

es inmediato comprobar que su envoltura convexa es de la forma

$$\mathrm{co}(\mathcal{O}) = \mathcal{O} \cup \left\{ (r\cos(\theta), r\,\mathrm{sen}(\theta)) : \pi/4 \le \theta \le 3\pi/4,\ 0 \le r \le \frac{\sqrt{2}}{2\,\mathrm{sen}(\theta)} \right\}$$

Es decir, debe añadirse al conjunto $\mathcal{O}$ el triángulo de la parte superior para obtener el menor convexo que lo contiene (Figura 1.2).



Conjunto no convexo $\mathcal{O}$ Envoltura convexa de $\mathcal{O}$

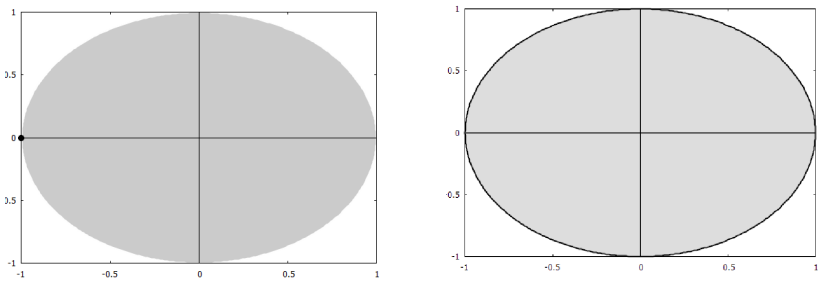
Figura 1.2: Convexificación

Ejemplo 1.4 El conjunto $C = \left\{(x,y) \in \mathbb{R}^2 : x^2 + y^2 < 1\right\} \cup \{(-1,0)\}$ es convexo, luego $\mathrm{co}(C) = C$ (Figura 1.3). Sin embargo no es cerrado, por lo que $\overline{\mathrm{co}}(C) \neq \mathrm{co}(C)$. Claramente la envoltura convexa cerrada de C es la bola unidad cerrada, $\overline{\mathrm{co}}(C) = \mathcal{B}$.

Dada una familia finita arbitraria de vectores $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^N$, una combinación lineal $\lambda_1\mathbf{x}_1 + \cdots + \lambda_m\mathbf{x}_m$ se dice que es una

- *Combinación convexa* si $\lambda_1 + \cdots + \lambda_m = 1$, con $0 \le \lambda_i \le 1$.
- *Combinación afín* si $\lambda_1 + \cdots + \lambda_m = 0$.

Si un conjunto es convexo debe contener a todas las combinaciones convexas de sus elementos. La prueba de esta afirmación es un ejemplo de utilización



$\text{co}(C) = C$ $\qquad$ $\overline{\text{co}}(C) = \mathcal{B}$

Figura 1.3: Envoltura convexa cerrada

del principio de inducción. En efecto, si $m = 2$ es evidente de la definición de convexidad. Supongamos que C es convexo y contiene las combinaciones convexas de $m-1$ elementos. Si tomamos una de m elementos

$$\sum_{i=1}^{m} \lambda_i \mathbf{x}_i = \lambda_1 \mathbf{x}_1 + (1-\lambda_1) \underbrace{\left(\sum_{i=2}^{m} \frac{\lambda_i}{(1-\lambda_1)} \mathbf{x}_i \right)}_{\mathbf{y}} \tag{1.10}$$

teniendo en cuenta que $\sum_{i=2}^{m} \frac{\lambda_i}{(1-\lambda_1)} = \frac{1-\lambda_1}{1-\lambda_1} = 1$ y la hipótesis de inducción, $\mathbf{y} \in C$, luego la combinación anterior estará en C, ya que hemos conseguido escribirla como combinación convexa de dos elementos del conjunto.

La siguiente proposición caracteriza la envoltura convexa en términos de combinaciones convexas.

Proposición 1.1 *Dado un conjunto $C \subset \mathbb{R}^N$, se tiene que* $\text{co}(C)$ *es la familia de todas las combinaciones convexas de elementos de C.*

Una familia de vectores $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_m$ se dice que es *afínmente independiente* si la única combinación afín que permite obtener el cero es la que tiene todos los escalares nulos, es decir, si

$$\lambda_0 \mathbf{x}_0 + \lambda_1 \mathbf{x}_1 + \cdots + \lambda_m \mathbf{x}_m = \mathbf{0}$$

con $\sum_{j=0}^{m} \lambda_j = 0$, implica necesariamente que $\lambda_j = 0$, para cada $0 \leq j \leq m$. Se comprueba fácilmente que los vectores $\{\mathbf{x}_i\}_{i=0}^{m}$ son afínmente independiente si y solamente si los vectores $\{\mathbf{x}_i - \mathbf{x}_0\}_{i=1}^{m}$ son linealmente independientes, por lo que en $\mathbb{R}^N$ las familias afínmente independientes tienen a lo sumo $N+1$ elementos.

Se llama *p-simplex* a la envoltura convexa de una familia de $p+1$ vectores afínmente independientes, que será de la forma

$$\text{co}\,(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_p) = \left\{ \sum_{i=0}^{p} \lambda_i \mathbf{x}_i : 0 \leq \lambda_i,\ \sum_{i=0}^{p} \lambda_i = 1 \right\} \tag{1.11}$$

Los puntos $\{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_p\}$ se denominan *vértices* del p-simplex.

Ejemplo 1.5 Dados tres vectores afínmente independientes $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2\}$ en $\mathbb{R}^N$, $N \geq 2$, el 2-simplex asociado es el triángulo de vértices $\mathbf{x}_0$, $\mathbf{x}_1$ y $\mathbf{x}_2$. En particular, si tomamos los puntos $\{(0,0), (1,1), (2,-1)\}$ en $\mathbb{R}^2$ se obtiene el 2-simplex de la Figura 1.4. En el caso de dos y cuatro vectores los simplex correspondientes son segmentos y tetraedros, respectivamente.

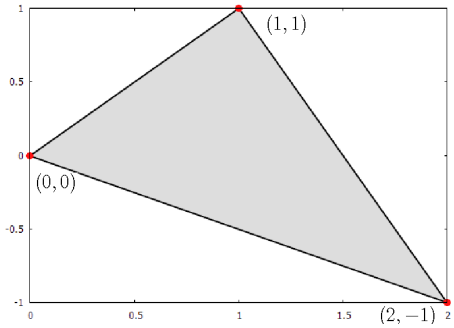


Figura 1.4: 2-simplex

Utilizando las nociones anteriores se obtiene una versión mejorada de la Proposición 1.1, en el sentido de que los elementos de la envoltura convexa de un conjunto de $\mathbb{R}^N$ son combinaciones convexas de $N+1$ elementos.

Teorema 1.1 (Carathéodory) *Dado un conjunto no vacío $C \subset \mathbb{R}^N$, se tiene que*

$$\mathrm{co}(C) = \left\{ \sum_{i=1}^{N+1} \lambda_i \mathbf{x}_i : 0 \leq \lambda_i, \sum_{i=1}^{N+1} \lambda_i = 1, \ \mathbf{x}_i \in C \right\} \tag{1.12}$$

El teorema de Carathéodory es un resultado muy importante con múltiples consecuencias relevantes, entre ellas la propiedad de que la envoltura convexa de un conjunto compacto (cerrado y acotado) de $\mathbb{R}^N$ es también un compacto.

Corolario 1.1 *Si $K \subset \mathbb{R}^N$ es un conjunto compacto, se tiene que* $\mathrm{co}(K)$ *es asimismo compacto.*

1.1.4 Hiperplano soporte

La noción de *hiperplano soporte* permite definir en la frontera de los conjuntos convexos un objeto similar al *plano tangente* de la geometría diferencial. De hecho, en los puntos en los que la frontera del conjunto convexo puede describirse localmente como una variedad diferenciable, el hiperplano soporte no es más que el plano tangente a dicha variedad en el punto.

Empezaremos con la definición formal de hiperplano. Dados $\mathbf{u} \in \mathbb{R}^N \setminus \{\mathbf{0}\}$, $\mathbf{x}_0 \in \mathbb{R}^N$, se llama *hiperplano afín* perpendicular a $\mathbf{u}$ conteniendo a $\mathbf{x}_0$ al conjunto

$$\mathcal{H} = \left\{\mathbf{x} \in \mathbb{R}^N : \langle \mathbf{u}, \mathbf{x} - \mathbf{x}_0 \rangle = 0\right\}$$

El hiperplano $\mathcal{H}$ permite separar el espacio $\mathbb{R}^N$ en dos semiplanos

$$\mathcal{H}^- = \left\{\mathbf{x} \in \mathbb{R}^N : \langle \mathbf{u}, \mathbf{x} - \mathbf{x}_0 \rangle \leq 0\right\} \qquad \mathcal{H}^+ = \left\{\mathbf{x} \in \mathbb{R}^N : \langle \mathbf{u}, \mathbf{x} - \mathbf{x}_0 \rangle \geq 0\right\}$$

Es decir, $\mathcal{H}^+ \cap \mathcal{H}^- = \mathcal{H}$ y $\mathbb{R}^N = \mathcal{H}^+ \cup \mathcal{H}^-$.

El siguiente resultado de tipo topológico resulta clave para poder definir el hiperplano soporte, ya que permite obtener dos teoremas denominados de *separación* de un conjunto convexo y un punto no contenido en el mismo.

Teorema 1.2 (Lema de accesibilidad) *Sea $C \subset \mathbb{R}^N$ un conjunto convexo de interior no vacío. Dados $\mathbf{x} \in \overline{C}$, $\mathbf{z} \in \operatorname{int}(C)$, se tiene que para cada $0 < \lambda \leq 1$,*

$$(1 - \lambda)\mathbf{x} + \lambda\mathbf{z} \in \operatorname{int}(C)$$

es decir, el segmento uniendo un punto del interior de C con otro punto arbitrario de su clausura está contenido en $\operatorname{int}(C)$.

Sean $C \subset \mathbb{R}^N$ un conjunto, $\mathbf{x} \notin C$ un punto y $\{\mathbf{z}_m\} \subset C$ una sucesión de aproximantes de la distancia $d_C(\mathbf{x})$ (ver página 6). De la desigualdad triangular

$$|\mathbf{z}_m| \leq |\mathbf{z}_m - \mathbf{x}| + |\mathbf{x}|$$

luego la sucesión está acotada y, como consecuencia del teorema de Heine-Borel, existe una subsucesión convergente que por simplicidad seguiremos denotando $\{\mathbf{z}_m\}$. Obviamente, si $\hat{\mathbf{z}}$ es el límite de la sucesión, se tiene que $\hat{\mathbf{z}} \in \overline{C}$ y $|\hat{\mathbf{z}} - \mathbf{x}| = d_C(\mathbf{x})$.

Por otra parte, si C es convexo y cerrado, para cada $\mathbf{z} \in C$, $0 < \lambda < 1$, se tiene que $(1 - \lambda)\hat{\mathbf{z}} + \lambda\mathbf{z} \in C$, luego

$$|\hat{\mathbf{z}}-\mathbf{x}|^2 \leq |(1-\lambda)\hat{\mathbf{z}}+\lambda\mathbf{z}-\mathbf{x}|^2 = (1-\lambda)^2|\hat{\mathbf{z}}-\mathbf{x}|^2+\lambda^2|\mathbf{z}-\mathbf{x}|^2+2(1-\lambda)\lambda\langle\hat{\mathbf{z}}-\mathbf{x}, \mathbf{z}-\mathbf{x}\rangle$$

Simplificando, se llega a la expresión

$$0 \leq (\lambda - 2)|\hat{\mathbf{z}} - \mathbf{x}|^2 + \lambda|\mathbf{z} - \mathbf{x}|^2 + 2(1 - \lambda)\langle\mathbf{z} - \mathbf{x}, \mathbf{z} - \mathbf{x}\rangle$$

y tomando límite cuando $\lambda \to 0$,

$$0 \leq -2\langle\hat{\mathbf{z}} - \mathbf{x}, \hat{\mathbf{z}} - \mathbf{x}\rangle + 2\langle\hat{\mathbf{z}} - \mathbf{x}, \mathbf{z} - \mathbf{x}\rangle = 2\langle\hat{\mathbf{z}} - \mathbf{x}, \mathbf{z} - \hat{\mathbf{z}}\rangle.$$

Hemos visto, pues, que si $C \subset \mathbb{R}^N$ es un convexo cerrado, para cada $\mathbf{x} \notin C$, existe $\hat{\mathbf{z}} \in C$ de forma que $|\hat{\mathbf{z}} - \mathbf{x}| = d_C(\mathbf{x})$, verificándose además la desigualdad variacional

$$\langle \mathbf{x} - \hat{\mathbf{z}},\, \mathbf{z} - \hat{\mathbf{z}}\rangle \leq 0, \qquad \forall\, \mathbf{z} \in C. \tag{1.13}$$

De (1.13) se deducen dos importantes consecuencias:

- Si $\hat{\mathbf{z}} \in C$ verifica la desigualdad variacional, entonces $|\hat{\mathbf{z}} - \mathbf{x}|$ proporciona la distancia de $\mathbf{x}$ a C. En efecto, dado $\mathbf{z} \in C$,

$$|\mathbf{z} - \mathbf{x}|^2 = |\mathbf{z} - \hat{\mathbf{z}}|^2 + 2\langle \mathbf{z} - \hat{\mathbf{z}}, \bar{\mathbf{z}} - \mathbf{x}\rangle + |\mathbf{z} - \hat{\mathbf{z}}|^2 \geq |\mathbf{z} - \hat{\mathbf{z}}|^2 + |\mathbf{z} - \hat{\mathbf{z}}|^2$$

  de donde

$$d_C(\mathbf{x}) \geq \inf_{\mathbf{z}\in C} \sqrt{|\mathbf{z} - \hat{\mathbf{z}}|^2 + |\mathbf{z} - \hat{\mathbf{z}}|^2} = |\hat{\mathbf{z}} - \mathbf{x}| = d_C(\mathbf{x})$$

- El punto donde se alcanza la distancia es único, ya que si existieran dos puntos distintos $\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2 \in C$, de (1.13)

$$\left.\begin{aligned} \langle \mathbf{x} - \hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2 - \hat{\mathbf{z}}_1\rangle \leq 0 \\ \langle \mathbf{x} - \hat{\mathbf{z}}_2, \hat{\mathbf{z}}_1 - \hat{\mathbf{z}}_2\rangle \leq 0 \end{aligned}\right\}$$

  y sumando ambas desigualdades de obtiene que $|\hat{\mathbf{z}}_2 - \hat{\mathbf{z}}_1|^2 \leq 0$, es decir, $\hat{\mathbf{z}}_1 = \hat{\mathbf{z}}_2$.

El elemento de C donde se alcanza la distancia al conjunto del punto $\mathbf{x}$ se denomina *proyección ortogonal* de $\mathbf{x}$ sobre C y se denota $\mathrm{proj}_C(\mathbf{x})$ (del inglés *projection*).

Proposición 1.2 *Sea $C \subset \mathbb{R}^N$ un conjunto convexo cerrado no vacío. Se tiene que el campo vectorial proyección ortogonal $\mathrm{proj}_C : \mathbb{R}^N \longrightarrow C$ es Lipschitz. Además, si $\mathbf{x} \notin C$, se tiene que $\mathrm{proj}_C(\mathbf{x}) \in \partial C$ (en otro caso $\mathrm{proj}_C(\mathbf{x}) = \mathbf{x}$).*

La existencia de proyección ortogonal sobre los conjuntos convexos cerrados y su caracterización variacional permiten obtener teoremas de separación para conjuntos convexos y puntos exteriores así como establecer la existencia de hiperplano soporte.

Teorema 1.3 (Separación de un convexo cerrado y un punto) *Sean un conjunto convexo y cerrado no vacío $C \subset \mathbb{R}^N$ y un punto $\mathbf{x}_0 \notin C$. Existirán $\mathbf{u}_0 \in \mathbb{R}^N \setminus \{\mathbf{0}\}$ y $\varepsilon > 0$, de forma que para cada $\mathbf{z} \in C$,*

$$\langle \mathbf{u}_0, \mathbf{z}\rangle \leq \langle \mathbf{u}_0, \mathbf{x}_0\rangle - \varepsilon, \tag{1.14}$$

Nota 1.1 Si $\mathcal{H}_0$ es el hiperplano asociado a $\mathbf{u}_0$ que pasa por $\mathbf{x}_0$, lo que afirma el teorema anterior es que $C \subset \operatorname{int} \mathcal{H}_0^-$. Este resultado no es cierto en general si se elimina la convexidad del conjunto. Si consideramos el conjunto $\mathcal{O}$ del Ejemplo 1.3, es inmediato ver que $(0, 0.5) \notin \mathcal{O}$ y, sin embargo, cualquier plano pasando por dicho punto separa el conjunto en dos mitades.

Nota 1.2 Dividiendo por $|\mathbf{u}_0|$ la desigualdad (1.14), es evidente que podemos suponer que el vector normal al hiperplano es unitario.

Teorema 1.4 (Separación de un convexo y un punto) *Sea $C \subset \mathbb{R}^N$ un conjunto convexo cuyo interior es no vacío. Para cada $\mathbf{x}_0 \notin C$ existe $\mathbf{u}_0 \in \mathbb{R}^N \setminus \{\mathbf{0}\}$ de forma que para cada $\mathbf{z} \in C$,*

$$\langle \mathbf{u}_0, \mathbf{z} \rangle \leq \langle \mathbf{u}_0, \mathbf{x}_0 \rangle.$$

Dado un conjunto no vacío $D \subset \mathbb{R}^N$, se denomina *función soporte* de D a la aplicación

$$\sigma_D(\mathbf{u}) = \sup_{\mathbf{x} \in D} \langle \mathbf{u}, \mathbf{x} \rangle. \tag{1.15}$$

Por convenio, si $D = \emptyset$ se toma σ_D constante e igual a $-\infty$. En otro caso, es evidente que $-\infty < \sigma_D(\mathbf{u}) \leq +\infty$ para cada $\mathbf{x} \in \mathbb{R}^N$. El dominio de la función soporte,

$$b(D) = \left\{ \mathbf{u} \in \mathbb{R}^N : \sigma_D(\mathbf{u}) \in \mathbb{R} \right\}$$

se denomina *cono barrera* (*barrier cone*) de D.

Teorema 1.5 (Existencia de hiperplano soporte) *Sea $C \subset \mathbb{R}^N$ convexo con interior no vacío. Para cada $\mathbf{x}_0 \in \partial C$ existe $\mathbf{u}_0 \in \mathbb{R}^N \setminus \{\mathbf{0}\}$ tal que*

$$\langle \mathbf{u}_0, \mathbf{x}_0 \rangle = \sigma_C(\mathbf{u}_0). \tag{1.16}$$

El hiperplano $\mathcal{H}_0 = \left\{ \mathbf{x} \in \mathbb{R}^N : \langle \mathbf{u}_0, \mathbf{x} - \mathbf{x}_0 \rangle = 0 \right\}$ se dice que *soporta* al conjunto C o que es el *hiperplano soporte* de C en el punto $\mathbf{x}_0 \in \partial C$. Obviamente dicho hiperplano contiene al punto $\mathbf{x}_0$ y además $C \subset \mathcal{H}_0^-$. Por otra parte, si $\mathbf{x}_0 + \mu \mathbf{u}_0$, se tiene que

$$\langle \mathbf{u}_0, \mathbf{x}_0 + \mu \mathbf{u}_0 \rangle = \langle \mathbf{u}_0, \mathbf{x}_0 \rangle + \mu |\mathbf{u}_0|^2$$

luego $\mathbf{x}_0 + \mu \mathbf{u}_0 \notin C$, si $\mu > 0$, lo que indica que el vector $\mathbf{u}_0$ *apunta hacia fuera* de C. Además, para cada $\mathbf{z} \in C$, de (1.16),

$$\langle \mathbf{z} - \mathbf{x}_0, \mathbf{x}_0 + \mu \mathbf{u}_0 - \mathbf{x}_0 \rangle = \mu \langle \mathbf{z} - \mathbf{x}_0, \mathbf{u}_0 \rangle \leq 0$$

lo que indica que $\mathrm{proj}_C(\mathbf{x}_0 + \mu \mathbf{u}_0) = \mathbf{x}_0$, es decir, la dirección $\mathbf{u}_0$ permite acercarse a C con velocidad de orden μ:

$$\lim_{\mu \to 0^+} \frac{d_C(\mathbf{x}_0 + \mu \mathbf{u}_0)}{\mu} = |\mathbf{u}_0|.$$

Recíprocamente, si $\mathrm{proj}_C(\mathbf{x}_0 + \mu \mathbf{u}) = \mathbf{x}_0$ para $\mu > 0$, de la caracterización variacional de la proyección orotogonal (1.13), es inmediato deducir que el hiperplano normal a $\mathbf{u}$ soporta al conjunto C en $\mathbf{x}_0$. Resumiendo,

Proposición 1.3 *Sea $C \subset \mathbb{R}^N$ con interior no vacío. Para cada $\mathbf{x}_0 \in \partial C$, se tiene que el hiperplano asociado al vector $\mathbf{u} \in \mathbb{R}^N \setminus \{\mathbf{0}\}$ soporta al conjunto en $\mathbf{x}_0$ si y solamente si $\mathrm{proj}_C(\mathbf{x}_0 + \mu \mathbf{u}) = \mathbf{x}_0$, $\mu > 0$.*

Ejemplo 1.6 La convexidad es esencial para garantizar la existencia de hiperplano soporte. Así, si consideramos el conjunto $\mathcal{O}$ del Ejemplo 1.3, es evidente que $(0,0) \in \partial\mathcal{O}$ y, sin embargo, no existe ningún plano soportando al conjunto en dicho punto, como se aprecia en la Figura 1.2.

Ejemplo 1.7 Consideremos el triángulo T de vértices $(0,0)$, $(1,0)$, $(0,1)$, que obviamente es de la forma

$$T = \operatorname{co}\{(0,0),\,(1,0),\,(0,1)\} = \left\{(x,y) \in \mathbb{R}^2 : 0 \le x, y \le 1,\ x+y \le 1\right\}.$$

Claramente $(0,0) \in \partial T$ y si tomamos el plano

$$\mathcal{H}_{(a,b)} = \left\{(x,y) \in \mathbb{R}^2 : ax + by = 0\right\}$$

con $a, b \le 0$, se tiene que $(0,0) \in \mathcal{H}_{(a,b)}$ y $T \subset \mathcal{H}^-_{(a,b)}$, lo que indica que se trata de un plano soporte del conjunto T en el origen. En la Figura 1.5 se muestran (en línea discontinua) los planos soporte al triángulo T en el origen para los vectores $\mathbf{u} = (-\sqrt{2}/4, -\sqrt{2}/4)$ y $\mathbf{v} = (-1/4, -1/2)$. Este ejemplo muestra que el hiperplano soporte de un convexo en un punto de la frontera no es necesariamente único.

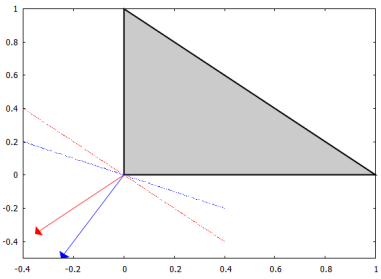


Figura 1.5: Hiperplanos soportando T

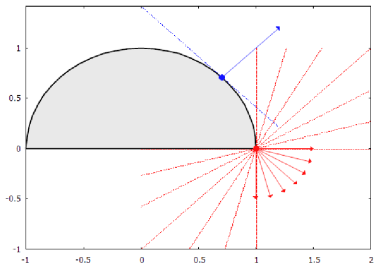


Figura 1.6: Hiperplanos soporte

Ejemplo 1.8 Sea $C \subset \mathbb{R}^N$ un conjunto convexo con interior no vacío y sea $\mathbf{x}_0 \in \partial C$ de forma que la frontera de C es una $(N-1)$-variedad diferenciable en un entorno, es decir, existen $U \subset \mathbb{R}^N$ un abierto conteniendo a $\mathbf{x}_0$ y una función $\varphi : U \longrightarrow \mathbb{R}$ de clase C^1 de forma que

(i) $\nabla\varphi(\mathbf{x}) \neq \mathbf{0}$, $\forall\ \mathbf{x} \in U$

(ii) $U \cap \partial C = \{\mathbf{x} \in U : \varphi(\mathbf{x}) = 0\}$

(iii) $U \cap C = \{\mathbf{x} \in U : \varphi(\mathbf{x}) \le 0\}$

Como consecuencia de **(i)** podemos suponer $\frac{\partial\varphi}{\partial x_N}(\mathbf{x}_0) \neq 0$ y del Teorema de la función implícita existen abiertos $V \subset \mathbb{R}^{N-1}$, $I \subset \mathbb{R}$, con $\mathbf{x}_0 \in V \times I \subset U$. También una función $\psi : V \longrightarrow I$ de clase C^1 tal que $\varphi(\mathbf{x}) = 0$ si y solamente

$x_N = \psi(x_1, \ldots, x_{N-1})$ para $\mathbf{x} = (x_1, \ldots, x_{N-1}, x_N) \in V \times I$. Tomemos ahora un vector unitario $\mathbf{u}$. Para $\mu > 0$ suficientemente pequeño, $\mathbf{x}_0 + \mu\mathbf{u} \in V \times I$, por lo que

$$d_C^2(\mathbf{x}_0 + \mu\mathbf{u}) = \inf_{\mathbf{y} \in V} \underbrace{|\mathbf{x}_0 + \mu\mathbf{u} - (\mathbf{y}, \psi(\mathbf{y}))|^2}_{f(\mathbf{y})}$$

Es inmediato comprobar que, para cada $1 \leq j \leq N-1$,

$$\frac{\partial f}{\partial y_j}(\mathbf{y}) = -2\left(\langle \mathbf{x}_0 + \mu\mathbf{u},\, \mathbf{e}_j \rangle - y_j\right) - 2\left(\langle \mathbf{x}_0 + \mu\mathbf{u},\, \mathbf{e}_N \rangle - \psi(\mathbf{y})\right) \frac{\partial \psi}{\partial y_j}(\mathbf{y}), \quad (1.17)$$

con $\{\mathbf{e}_1, \ldots, \mathbf{e}_N\}$ la base canónica de $\mathbb{R}^N$. De la Proposición 1.3, $\mathbf{u}$ generará un hiperplano soporte de C en $\mathbf{x}_0$ si y solamente si $\mathrm{proj}_C(\mathbf{x}_0 + \mu\mathbf{u}) = \mathbf{x}_0$, es decir, si $\mathbf{x}_0$ minimiza f o, equivalentemente, si es un punto crítico, lo que por las ecuaciones (1.17) nos lleva a escribir las condiciones

$$-2\mu u_j - 2\mu u_N \frac{\partial \psi}{\partial y_j}(\mathbf{y}_0) = 0 \;\Leftrightarrow\; u_j = -u_N \frac{\partial \psi}{\partial y_j}(\mathbf{y}_0), \qquad 1 \leq j \leq N-1 \quad (1.18)$$

$\mathbf{y}_0 = (x_{01}, \ldots, x_{0N-1}) \in \mathbb{R}^{N-1}$. Por otra parte, dado que $\varphi(\mathbf{y}, \psi(\mathbf{y})) = 0$, $\mathbf{y} \in V$, se tiene que para cada $1 \leq j \leq N-1$

$$\frac{\partial \varphi}{\partial x_j}(\mathbf{y}, \psi(\mathbf{y})) + \frac{\partial \varphi}{\partial x_N}(\mathbf{y}, \psi(\mathbf{y})) \frac{\partial \psi}{\partial y_j}(\mathbf{y}) = 0, \qquad \mathbf{y} \in V.$$

En particular, para cada $1 \leq j \leq N-1$

$$\frac{\partial \varphi}{\partial x_j}(\mathbf{x}_0) + \frac{\partial \varphi}{\partial x_N}(\mathbf{x}_0) \frac{\partial \psi}{\partial y_j}(\mathbf{y}_0) = 0 \quad (1.19)$$

Combinando (1.18) con (1.19) se llega a la relación

$$u_j = u_N \frac{\partial \varphi}{\partial x_j}(\mathbf{x}_0) / \frac{\partial \varphi}{\partial x_N}(\mathbf{x}_0) \quad (1.20)$$

que garantiza que el único vector unitario que genera un hiperplano soporte es $\nabla\varphi(\mathbf{x}_0)/|\nabla\varphi(\mathbf{x}_0)|$, siendo la ecuación del hiperplano

$$\mathcal{H} = \left\{\mathbf{x} \in \mathbb{R}^N : \langle \mathbf{x} - \mathbf{x}_0,\, \nabla\varphi(\mathbf{x}_0) \rangle = 0\right\}$$

que es el hiperplano tangente de la geometría diferencial clásica. Resumiendo, la noción de hiperplano soporte de un convexo en un punto de la frontera extiende la noción de hiperplano tangente a puntos en los que la frontera no es una variedad diferenciable (como ocurre en el ejemplo anterior). Además, en los puntos en los que la frontera del conjunto sí es una variedad diferenciable el hiperplano soporte es único y coincide con el tangente (ver Figura 1.6).

1.2 Funciones convexas

Aunque ya habían sido consideradas anteriormente, el estudio sistemático de las funciones convexas se inicia alrededor de los años sesenta del siglo pasado motivado fundamentalmente por problemas de optimización, R.T. Rockafellar, y mecánica no regular (*nonsmooth mechanics*), J.J. Moreau.

Resulta crucial en este campo la caracterización de la convexidad de una función en términos de su epigráfica, Proposición 1.4, lo que permite conectar el tratamiento analítico de las funciones con el estudio geométrico de los conjuntos convexos.

1.2.1 Conceptos básicos

Definición 1.1 *Dado un conjunto convexo $D \subseteq \mathbb{R}^N$, se dice que una función $\phi : D \longrightarrow \mathbb{R}$ es* convexa *si dados $\mathbf{x}_1, \ldots, \mathbf{x}_m \in D$, $\lambda_j \geq 0$, $1 \leq j \leq m$, con $\lambda_1 + \cdots + \lambda_m = 1$, se tiene que*

$$\phi\left(\sum_{j=1}^{m} \lambda_j \mathbf{x}_j\right) \leq \sum_{j=1}^{m} \lambda_j \phi(\mathbf{x}_j) \tag{1.21}$$

La expresión anterior se denomina desigualdad de Jensen.

Razonando como en el caso de los conjuntos convexos (ver (1.10) en la página 10) es sencillo comprobar que una función $\phi : D \subseteq \mathbb{R}^N \longrightarrow \mathbb{R}$ es convexa si y solamente si para cada par de vectores $\mathbf{x}, \mathbf{y} \in D$ y cada $0 \leq \lambda \leq 1$, se tiene la desigualdad

$$\phi((1-\lambda)\mathbf{x} + \lambda\mathbf{y}) \leq (1-\lambda)\phi(\mathbf{x}) + \lambda\phi(\mathbf{y})$$

Asociados a una función arbitraria $f : D \subseteq \mathbb{R}^N \longrightarrow \mathbb{R}$ se definen los siguientes subconjuntos de $\mathbb{R}^{N+1}$:

- *Gráfica* o *grafo* de f

$$\operatorname{graph}(f) = \{(\mathbf{x}, f(\mathbf{x})) : \mathbf{x} \in D\} \tag{1.22}$$

- *Epigráfica* o *epigrafo* de f

$$\operatorname{epi}(f) = \left\{(\mathbf{x}, \alpha) \in \mathbb{R}^{N+1} : \mathbf{x} \in D,\ f(\mathbf{x}) \leq \alpha\right\} \tag{1.23}$$

Claramente $\operatorname{graph}(f) \subset \operatorname{epi}(f)$, además es evidente que la epigráfica de una función es el conjunto de los puntos de $\mathbb{R}^{N+1}$ que están "por encima" de la gráfica.

Los conjuntos anteriores permiten dar una interpretación geométrica de la noción de función convexa. En efecto, la convexidad de una función ϕ equivale a

que dados dos puntos arbitrarios en su gráfica, $(\mathbf{x}, \phi(\mathbf{x}))$, $(\mathbf{y}, \phi(\mathbf{y}))$, el segmento que los une esté en $\mathrm{epi}(\phi)$, es decir, que para cada $0 \leq \lambda \leq 1$,

$$(1-\lambda)(\mathbf{x}, \phi(\mathbf{x}))+\lambda(\mathbf{y}, \phi(\mathbf{y})) \in \mathrm{epi}(\phi) \;\Leftrightarrow\; \phi((1-\lambda)\mathbf{x}+\lambda\mathbf{y}) \leq (1-\lambda)\phi(\mathbf{x})+\lambda\phi(\mathbf{y})$$

lo cual significa que el segmento uniendo dos puntos cualesquiera de la gráfica de una función convexa está siempre por encima de la gráfica.

Ejemplo 1.9 Consideremos la función $\phi(\mathbf{x}) = \frac{1}{2}|\mathbf{x}|^2$, $\mathbf{x} \in \mathbb{R}^N$. Esta función es convexa, dado que, de la definición de norma euclídea de un vector

$$\begin{aligned} \frac{1}{2}\left|(1-\lambda)\mathbf{x}+\lambda\mathbf{y}\right|^2 &= \frac{1}{2}\left((1-\lambda)^2|\mathbf{x}|^2 + 2(1-\lambda)\lambda\langle\mathbf{x},\mathbf{y}\rangle + \lambda^2|\mathbf{y}|^2\right) \\ &\leq \frac{1}{2}\left((1-\lambda)^2|\mathbf{x}|^2 + 2(1-\lambda)\lambda|\mathbf{x}|\,|\mathbf{y}| + \lambda^2|\mathbf{y}|^2\right) \\ &= \frac{1}{2}\left((1-\lambda)|\mathbf{x}| + \lambda|\mathbf{y}|\right)^2 \end{aligned} \tag{1.24}$$

usando además la desigualdad de Cauchy-Schwarz. Por otra parte, dados dos números reales $a, b > 0$, de la fórmula del cuadrado del binomio se tiene que $(a+b)^2 \leq a^2 + b^2$ y si $0 \leq \lambda \leq 1$,

$$\left((1-\lambda)a+\lambda b\right)^2 \leq (1-\lambda)^2a^2 + \lambda^2b^2 \leq (1-\lambda)a^2 + \lambda b^2 \tag{1.25}$$

Combinando las desigualdades anteriores se obtiene

$$\frac{1}{2}\left|(1-\lambda)\mathbf{x}+\lambda\mathbf{y}\right|^2 \leq (1-\lambda)\frac{|\mathbf{x}|^2}{2} + \lambda\frac{|\mathbf{y}|^2}{2}$$

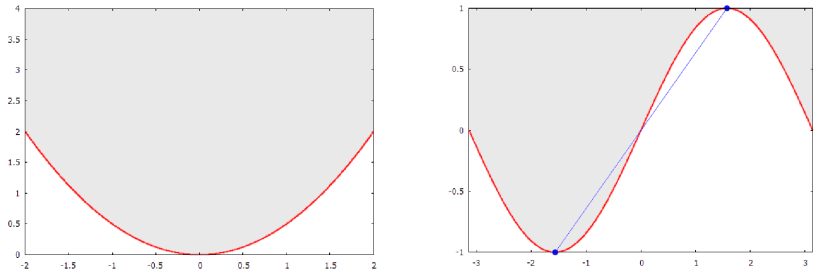
que proporciona la convexidad buscada. El dibujo de la izquierda de la Figura 1.7 corresponde a la epigráfica de la función anterior para $N = 1$. Se observa que el segmento uniendo dos puntos cualesquiera de la gráfica de ϕ (en rojo) está contenido en $\mathrm{epi}(\phi)$. Por el contrario, si tomamos la función $\varphi(x) = \mathrm{sen}(x)$, $x \in [-\pi, \pi]$, cuya epigráfica se ha representado también en la Figura 1.7, se observa que tomando, por ejemplo, los puntos $(-\pi/2, -1)$, $(\pi/2, 1) \in \mathrm{graph}(\varphi)$, el segmento que los une no está contenido en $\mathrm{epi}(\varphi)$, por lo que esta función no es convexa.

Ejemplo 1.10 (Formas cuadráticas) Sea Q una matriz cuadrada de tamaño $N \times N$, simétrica (es decir, de forma que coincide con su *traspuesta*, $Q = Q^t$) y semidefinida positiva, $\mathbf{x}^tQ\mathbf{x} \geq 0$, para cada $\mathbf{x} \in \mathbb{R}^N$. Con estas hipótesis se verifica la *desigualdad de Cauchy-Schwarz* que establece que

$$\forall\ \mathbf{x}, \mathbf{y} \in \mathbb{R}^N \qquad \left|\mathbf{x}^tQ\mathbf{y}\right|^2 \leq \left(\mathbf{x}^tQ\mathbf{x}\right)\left(\mathbf{y}^tQ\mathbf{y}\right) \tag{1.26}$$

Además, si definimos la función

$$\varphi(\mathbf{x}) = \mathbf{x}^tQ\mathbf{x} = \langle\mathbf{x}, Q\mathbf{x}\rangle, \qquad \mathbf{x} \in \mathbb{R}^N \tag{1.27}$$



$\phi(x) = \frac{x^2}{2}$ $\varphi(x) = \text{sen}(x)$

Figura 1.7: Epigráficas

se tiene que es convexa. En efecto, dados dos vectores $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$ y una constante $0 \leq \lambda \leq 1$, se tiene que

$$\begin{aligned} \varphi((1-\lambda)\mathbf{x} + \lambda\mathbf{y}) &= (1-\lambda)^2 \mathbf{x}^t Q\mathbf{x} + \lambda(1-\lambda)\left(\mathbf{x}^t Q\mathbf{y} + \mathbf{y}^t Q\mathbf{x}\right) + \lambda^2 \mathbf{y}^t Q\mathbf{y} \\ &= (1-\lambda)^2 \mathbf{x}^t Q\mathbf{x} + 2\lambda(1-\lambda)\mathbf{x}^t Q\mathbf{y} + \lambda^2 \mathbf{y}^t Q\mathbf{y} \end{aligned}$$

teniendo en cuenta que, por ser Q simétrica, $\mathbf{y}^t Q\mathbf{x} = \mathbf{x}^t Q^t \mathbf{y} = \mathbf{x}^t Q\mathbf{y}$. Finalmente, usando las desigualdades de Cauchy-Schwarz para formas cuadráticas arbitrarias (1.26) y (1.25), de la identidad anterior se obtiene la desigualdad de Jensen para φ:

$$\begin{aligned} \varphi((1-\lambda)\mathbf{x} + \lambda\mathbf{y}) &\leq (1-\lambda)^2 \mathbf{x}^t Q\mathbf{x} + 2\lambda(1-\lambda)\left(\mathbf{x}^t Q\mathbf{x}\right)^{1/2}\left(\mathbf{y}^t Q\mathbf{y}\right)^{1/2} + \lambda^2 \mathbf{y}^t Q\mathbf{y} \\ &= \left((1-\lambda)\left(\mathbf{x}^t Q\mathbf{x}\right)^{1/2} + \lambda\left(\mathbf{x}^t Q\mathbf{x}\right)^{1/2}\right)^2 \\ &\leq (1-\lambda)\left(\mathbf{x}^t Q\mathbf{x}\right) + \lambda\left(\mathbf{y}^t Q\mathbf{y}\right) = (1-\lambda)\varphi(\mathbf{x}) + \lambda\varphi(\mathbf{y}) \end{aligned}$$

Esta clase de funciones convexas se denominan *formas cuadráticas*.

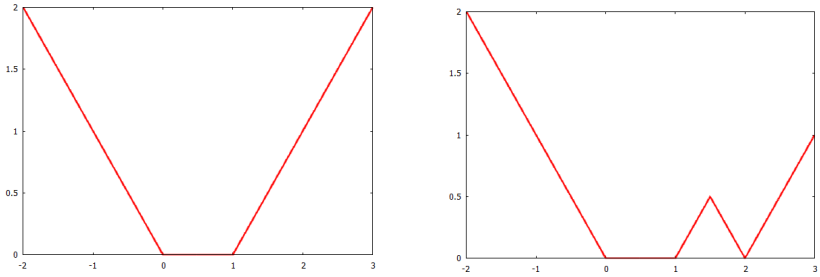
Ejemplo 1.11 (Funciones distancia) Dado un conjunto convexo $C \subset \mathbb{R}^N$, la función distancia a C, $d_C(\cdot)$, definida por la relación (1.4) es convexa. En efecto, dados $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$, para cada $\varepsilon > 0$ existirán $\mathbf{x}_\varepsilon, \mathbf{y}_\varepsilon \in C$, de forma que

$$d_C(\mathbf{x}) + \varepsilon > |\mathbf{x} - \mathbf{x}_\varepsilon|, \qquad d_C(\mathbf{y}) + \varepsilon > |\mathbf{y} - \mathbf{y}_\varepsilon|$$

Por otra parte, si $0 \leq \lambda \leq 1$, al ser C convexo se tiene que $(1-\lambda)\mathbf{x}_\varepsilon + \lambda\mathbf{y}_\varepsilon \in C$, de donde

$$\begin{aligned} d_C((1-\lambda)\mathbf{x} + \lambda\mathbf{y}) &\leq |(1-\lambda)\mathbf{x} + \lambda\mathbf{y} - (1-\lambda)\mathbf{x}_\varepsilon - \lambda\mathbf{y}_\varepsilon| \\ &\leq (1-\lambda)|\mathbf{x} - \mathbf{x}_\varepsilon| + \lambda|\mathbf{y} - \mathbf{y}_\varepsilon| \\ &\leq (1-\lambda)\left(d_C(\mathbf{x}) + \varepsilon\right) + \lambda\left(d_C(\mathbf{y}) + \varepsilon\right) \\ &= (1-\lambda)d_C(\mathbf{x}) + \lambda d_C(\mathbf{y}) + \varepsilon \end{aligned}$$

Dado que la desigualdad anterior es válida para cada $\varepsilon > 0$, se tiene que la función distancia verifica la desigualdad de Jensen y, por tanto, es convexa. En la siguiente figura se muestran las gráficas de las funciones distancia a los conjuntos $I = [0,1]$ y $J = [0,1] \cup \{2\}$ mientras que la Figura 1.9 corresponde a la gráfica de la función



Gráfica de $d_I(\cdot)$ Gráfica de $d_J(\cdot)$

Figura 1.8: Funciones distancia

distancia al conjunto $D = \left\{(x,y) \in \mathbb{R}^2 : (x-1)^2 + (y-1)^2 \leq 1\right\}$. El recíproco

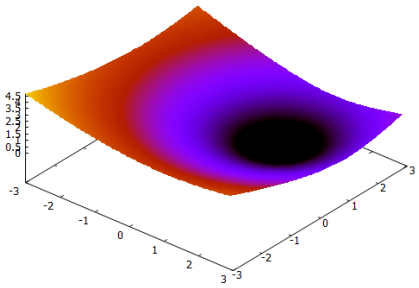


Figura 1.9: Gráfica de $d_D(\cdot)$

de la afirmación anterior también es cierto si $C \subset \mathbb{R}^N$ es un conjunto cerrado. En efecto, en ese caso sabemos que $C = \left\{\mathbf{x} \in \mathbb{R}^N : d_C(\mathbf{x}) = 0\right\}$, luego dados $\mathbf{x}, \mathbf{y} \in C$, si la función distancia es convexa, se tiene que

$$d_C((1-\lambda)\mathbf{x} + \lambda\mathbf{y}) \leq (1-\lambda)d_C(\mathbf{x}) + \lambda d_C(\mathbf{y}) = 0 \;\Leftrightarrow\; d_C((1-\lambda)\mathbf{x} + \lambda\mathbf{y}) = 0$$

para cada $0 \leq \lambda \leq 1$, es decir, $(1-\lambda)\mathbf{x} + \lambda\mathbf{y} \in C$.

La convexidad de una función puede caracterizarse en términos de la convexidad de su epigráfica.

Proposición 1.4 *Sea $D \subseteq \mathbb{R}^N$ un conjunto convexo. Se tiene que una función $\phi : D \longrightarrow \mathbb{R}$ es convexa si y solamente si $\operatorname{epi}(\phi) \subset \mathbb{R}^{N+1}$ es un conjunto convexo.*

La Proposición 1.4 proporciona una definición alternativa de función convexa. Así podemos llamar convexas a aquellas funciones cuya epigráfica es un conjunto convexo. Esto es especialmente útil para evitar problemas que pueden surgir en el término de la derecha de la desigualdad de Jensen, por ejemplo cuando se manejan funciones que toman valores en $\mathbb{R}\cup\{+\infty\}$, lo que es habitual, por ejemplo, al considerar problemas de optimización con restricciones (ver pág. 23).

Consideraremos desde este momento funciones $\phi : \mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$ definidas en todo $\mathbb{R}^N$ y que pueden tomar el valor $+\infty$ en algún punto y definimos su *dominio* como el conjunto

$$\mathrm{dom}(\phi) = \left\{\mathbf{x} \in \mathbb{R}^N : \phi(\mathbf{x}) \in \mathbb{R}\right\}. \tag{1.28}$$

La Proposición 1.4 proporciona así mismo un procedimiento sencillo y práctico para comprobar la convexidad de una función. En particular el siguiente corolario, que se deduce de la identidad

$$\mathrm{epi}\left(\sup_{i\in\mathcal{I}} \phi_i\right) = \bigcap_{i\in\mathcal{I}} \mathrm{epi}(\phi_i) \tag{1.29}$$

permite construir funciones convexas no necesariamente diferenciables a partir de otras que sí lo son.

Corolario 1.2 *Sea $\phi_i : \mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$ una familia de funciones convexas, $i \in \mathcal{I}$. Se tiene que la función*

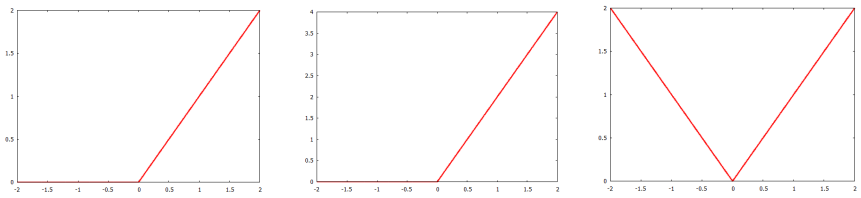
$$\phi(\mathbf{x}) = \sup_{i\in\mathcal{I}} \phi_i(\mathbf{x})$$

es convexa.

Ejemplo 1.12 (Funciones soporte) Del corolario anterior es evidente que las funciones soporte σ_D, definidas en (1.15), son convexas independientemente del conjunto D, al ser el supremo de una familia de funciones afines (y por tanto convexas). Así, si consideramos los conjuntos $I = [0,1]$ y $J = [0,1]\cup\{2\}$ es inmediato comprobar que

$$\sigma_I(u) = \begin{cases} u, & u \geq 0 \\ 0, & u < 0 \end{cases} \qquad \sigma_J(u) = \begin{cases} 2u, & u \geq 0 \\ 0, & u < 0 \end{cases}$$

mientras que $\sigma_{[0,1]\cup\{-1\}}(u) = |u|$.



σ_I σ_J $\sigma_{[0,1]\cup\{-1\}}$

Figura 1.10: Gráficas de funciones soporte

Además, en el caso de conjuntos no acotados, las funciones soporte porporcionan ejemplos de funciones que de forma natural toman el valor $+\infty$ en algunos puntos. En efecto, dado un conjunto $D \subset \mathbb{R}^N$, de forma que para un cierto vector unitario $\mathbf{u} \in \mathbb{R}^N$, $|\mathbf{u}| = 1$, y un punto $\mathbf{x} \in D$, se tiene una sucesión $\mu_m \to +\infty$ con $\mathbf{x} + \mu_m \mathbf{u}$, de la definición de función soporte

$$\sigma_D(\mathbf{u}) \geq \langle \mathbf{u}, \mathbf{x} \rangle + \mu_m \underset{m\to\infty}{\longrightarrow} +\infty$$

luego $\sigma_D(\mathbf{u}) = +\infty$.

1.2.2 Funciones convexas y diferenciables

Sea $\phi : \mathbb{R}^N \longrightarrow \mathbb{R} \cup \{+\infty\}$ una función, de forma que $\operatorname{dom}\phi \subset \mathbb{R}^N$ es un abierto convexo. Si ϕ es diferenciable se dispone de diversas caracterizaciones que resultan muy útiles para verificar si es o no convexa.

Proposición 1.5 *Sea $\phi : \mathbb{R}^N \longrightarrow \mathbb{R} \cup \{+\infty\}$ una función de clase C^1 en el abierto convexo $\operatorname{dom}\phi$. Se tiene que ϕ es convexa si y solamente si para cada $\mathbf{x}, \mathbf{y} \in D$ se verifica alguna de las siguientes desigualdades:*

$$\langle \nabla\phi(\mathbf{x}),\, \mathbf{y} - \mathbf{x} \rangle \leq \phi(\mathbf{y}) - \phi(\mathbf{x}) \tag{1.30}$$

$$\langle \nabla\phi(\mathbf{y}) - \nabla\phi(\mathbf{x}),\, \mathbf{y} - \mathbf{x} \rangle \geq 0 \tag{1.31}$$

Cuando se verifica (1.31) se dice que el gradiente de ϕ es *monótono*. En el caso unidimensional esto significa que la función derivada ϕ' es creciente.

Proposición 1.6 *Sea $\phi : \mathbb{R}^N \longrightarrow \mathbb{R} \cup \{+\infty\}$ una función de clase C^2 en el abierto convexo $\operatorname{dom}\phi$. Se tiene que ϕ es convexa si y solamente si la matriz Hessiana $\nabla^2\phi(\mathbf{x})$ es semidefinida positiva para cada $\mathbf{x} \in D$, es decir, si $\mathbf{y}^t \nabla^2\phi(\mathbf{x})\mathbf{y} \geq 0$, $\forall\ \mathbf{y} \in \mathbb{R}^N$.*

Al ser la matriz Hessiana simétrica, es diagonalizable, por lo que para cada $\mathbf{x} \in D$, existen escalares $\mu_1(\mathbf{x}), \ldots, \mu_N(\mathbf{x})$ (los *valores propios* de la matriz) y una matriz ortogonal $T(\mathbf{x})$ de forma que

$$\nabla^2\phi(\mathbf{x}) = T(\mathbf{x}) \begin{pmatrix} \mu_1(\mathbf{x}) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mu_N(\mathbf{x}) \end{pmatrix} T(\mathbf{x})^t$$

por tanto, para verificar si es semidefinida positiva, es suficiente con determinar si los valores propios son no negativos.

1.2.3 Propiedades de minimización

Dada una función $f : \mathbb{R}^N \longrightarrow \mathbb{R}$, el problema *minimizar* o *calcular el mínimo* de f en el conjunto $K \subset \mathbb{R}^N$, consiste en encontrar algún punto $\bar{\mathbf{x}}$ de forma que

$$\begin{aligned} &\text{(i) } \bar{\mathbf{x}} \in K \\ &\text{(ii) } f(\bar{\mathbf{x}}) = \inf_{\mathbf{x}\in K} f(\mathbf{x}) \end{aligned} \tag{1.32}$$

Definiendo la *función indicatriz* del conjunto

$$\psi_K(\mathbf{x}) = \begin{cases} 0, & \text{si } \mathbf{x} \in K \\ +\infty, & \text{si } \mathbf{x} \notin K \end{cases} \tag{1.33}$$

el problema (1.32) de minimizar f en K equivale al problema de minimizar la función $f_K = f + \psi_K : \mathbb{R}^N \longrightarrow \mathbb{R} \cup \{+\infty\}$ en todo el espacio $\mathbb{R}^N$. Por tanto, la utilización de funciones tomando el valor $+\infty$ permite reescribir los problemas de minimización con restricciones en forma de problemas de minimización global.

Entre las buenas propiedades de las funciones convexas destaca su buen comportamiento frente al cálculo de mínimos. Se enuncian seguidamente alguna de estas propiedades.

Proposición 1.7 (Convexidad de la función marginal) *Sea $f : D \times Q \subset \mathbb{R}^{N+M} \longrightarrow \mathbb{R}$ una función convexa inferiormente acotada. Si definimos la función $\phi : D \subset \mathbb{R}^N \longrightarrow \mathbb{R}$, de forma que*

$$\phi(\mathbf{x}) = \inf_{\mathbf{y}\in Q} f(\mathbf{x}, \mathbf{y})$$

se tiene que ϕ es convexa.

La convexidad de una función no es suficiente para garantizar la existencia de mínimo como muestra el Ejemplo 1.13. Sin embargo cuando existen mínimos el conjunto de todos ellos es convexo, lo que, en particular, impide que una función convexa tenga varios mínimos aislados.

Ejemplo 1.13 Consideremos la función $\phi : \mathbb{R} \longrightarrow \mathbb{R} \cup \{+\infty\}$ definida como

$$\phi(x) = \begin{cases} 1/x, & x > 0 \\ +\infty, & x \leq 0 \end{cases}$$

que claramente es convexa y propia, $\mathrm{dom}(\phi) =]0, +\infty[$. Además, el ínfimo de ϕ es igual a cero (basta tomar la sucesión $n \to +\infty$, que verifica $\phi(n) = 1/n \to 0$), pero $\phi(x) > 0$, para cada $x \in \mathbb{R}$.

Proposición 1.8 *Sea* $\phi : \mathbb{R}^N \longrightarrow \mathbb{R} \cup \{+\infty\}$ *una función convexa y estricta* ($\mathrm{dom}(\phi) \neq \emptyset$). *Se tiene que el conjunto de sus puntos mínimos,*

$$\operatorname{argmin} \phi = \left\{ \mathbf{x} \in \mathbb{R}^N : \phi(\mathbf{x}) = \inf_{\mathbf{y} \in \mathbb{R}^N} \phi(\mathbf{y}) \right\}$$

es convexo.

El conjunto $\operatorname{argmin} \phi$ se reduce a un punto, cuando es no vacío, si la función convexa ϕ es *estrictamente convexa* es decir, si dados $\mathbf{x}, \mathbf{y} \in \mathrm{dom}(\phi)$ dos puntos distintos se tiene que

$$\phi\left(\frac{\mathbf{x}+\mathbf{y}}{2}\right) < \frac{\phi(\mathbf{x}) + \phi(\mathbf{y})}{2} \tag{1.34}$$

En efecto, si $\overline{m}$ es el ínfimo de la función ϕ, dados dos puntos distintos $\mathbf{x}, \mathbf{y}$ en $\operatorname{argmin} \phi$, al ser este conjunto convexo (Proposición 1.8) se tiene que $(\mathbf{x}+\mathbf{y})/2 \in \operatorname{argmin} \phi$, luego

$$\overline{m} = \phi\left(\frac{\mathbf{x}+\mathbf{y}}{2}\right) < \frac{\phi(\mathbf{x}) + \phi(\mathbf{y})}{2} = \overline{m}$$

lo cual es absurdo. Cabe decir que en el caso de funciones diferenciables la convexidad estricta, al igual que la convexidad, puede caracterizarse en términos de las derivadas.

Proposición 1.9 *Sea* $\phi : \mathbb{R}^N \longrightarrow \mathbb{R} \cup \{+\infty\}$ *una función de clase* C^1 *en el abierto convexo* D. *Se tiene que* ϕ *es estrictamente convexa si y solamente si la desigualdad (1.30) o (1.31) se verifica de forma estricta para cada* $\mathbf{x} \neq \mathbf{y}$. *Si* ϕ *es* C^2 *una condición suficiente, aunque no necesaria, para la convexidad estricta de* ϕ *es que su matriz Hessiana sea definida positiva, es decir, para cada* $\mathbf{x} \in D$, $\mathbf{y}^t \nabla^2 \phi(\mathbf{x}) \mathbf{y} > 0$, $\forall\ \mathbf{y} \in \mathbb{R}^N \setminus \{\mathbf{0}\}$.

Como indica la proposición anterior, la positividad de la matriz Hessiana es una condición suficiente para la convexidad estricta de una función dos veces derivable, aunque no es necesaria. Si consideramos, por ejemplo, la función $g(x) = x^4$, se tiene que para cada $x, y \in \mathbb{R}$, $x \neq y$,

$$(g'(y) - g'(x))\,(y - x) = 4\left(y^3 - x^3\right)(y - x) = 4(y - x)^2\left(y^2 + xy + x^2\right) > 0$$

ya que, si $x \neq 0$, entonces $y^2 + xy + x^2 = \frac{3}{4}x^2 + \left(\frac{1}{2}x + y\right)^2 > 0$ y si $x = 0$, la expresión anterior queda reducida a $y^2 > 0$. Así, pues, g es una función estrictamente convexa. Sin embargo $g''(x) = 12x^2$ no es estrictamente positiva ya que $g''(0) = 0$.

Se dice que $\bar{\mathbf{x}}$ es un *mínimo local* de la función $f : \mathbb{R}^N \longrightarrow \mathbb{R} \cup \{+\infty\}$ si existe $\delta > 0$ de forma que, para cada $\mathbf{x} \in B_\delta(\bar{\mathbf{x}})$, se tiene $f(\bar{\mathbf{x}}) \leq f(\mathbf{x})$.

Proposición 1.10 *Sea* $\phi : \mathbb{R}^N \longrightarrow \mathbb{R} \cup \{+\infty\}$ *una función convexa y extricta,* $\mathrm{dom}(\phi) \neq \emptyset$. *Si* $\bar{\mathbf{x}}$ *es un mínimo local de* ϕ, *se tiene que* $\bar{\mathbf{x}} \in \operatorname{argmin} \phi$, *es decir, se trata de un mínimo (global) de la función.*

1.2.4 Envoltura de Moreau I

Sea $\phi : \mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$ una función convexa propia. Para cada $\lambda > 0$ se considera el problema de minimización

$$\phi_\lambda(\mathbf{x}) = \inf_{\mathbf{y}\in\mathbb{R}^N}\left(\phi(\mathbf{y}) + \frac{1}{2\lambda}|\mathbf{y}-\mathbf{x}|^2\right). \tag{1.35}$$

La aplicación $\phi_\lambda \leq \phi$, se denomina *envoltura de Moreau (Moreau envelope)*. Se considera también el problema

$$\operatorname*{argmin}_{\mathbf{y}\in\mathbb{R}^N}\left(\phi(\mathbf{y}) + \frac{1}{2\lambda}|\mathbf{y}-\mathbf{x}|^2\right) \tag{1.36}$$

de obtener los puntos en los que se alcanza el valor de $\phi_\lambda(\mathbf{x})$.

Dado $\mathbf{x} \in \mathrm{dom}(\phi)$, supongamos que $\bar{\mathbf{x}} \in \mathbb{R}^N$ es solución de (1.36), es decir, para cada $\mathbf{y} \in \mathbb{R}^N$

$$\phi(\bar{\mathbf{x}}) + \frac{1}{2\lambda}|\bar{\mathbf{x}}-\mathbf{x}|^2 \leq \phi(\mathbf{y}) + \frac{1}{2\lambda}|\mathbf{y}-\mathbf{x}|^2. \tag{1.37}$$

Tomando, en particular, el punto $(1-\theta)\bar{\mathbf{x}}+\theta\mathbf{y}$, con $0<\theta<1$, en la desigualdad anterior se obtiene

$$\begin{aligned}\phi(\bar{\mathbf{x}}) + \frac{1}{2\lambda}|\bar{\mathbf{x}}-\mathbf{x}|^2 &\leq \phi((1-\theta)\bar{\mathbf{x}}+\theta\mathbf{y}) + \frac{1}{2\lambda}|(1-\theta)\bar{\mathbf{x}}+\theta\mathbf{y}-\mathbf{x}|^2\\ &\leq (1-\theta)\phi(\bar{\mathbf{x}}) + \theta\phi(\mathbf{y}) + \frac{1}{2\lambda}\left(|\bar{\mathbf{x}}-\mathbf{x}|^2 + 2\theta\langle\bar{\mathbf{x}}-\mathbf{x}, \mathbf{y}-\bar{\mathbf{x}}\rangle + \theta^2|\mathbf{y}-\bar{\mathbf{x}}|^2\right)\end{aligned}$$

usando la convexidad de ϕ y las propiedades del producto escalar. Simplificando esta expresión se llega a

$$0 \leq \phi(\mathbf{y}) - \phi(\bar{\mathbf{x}}) + \frac{1}{\lambda}\langle\bar{\mathbf{x}}-\mathbf{x},\, \mathbf{y}-\bar{\mathbf{x}}\rangle + \frac{\theta}{2\lambda}|\mathbf{y}-\bar{\mathbf{x}}|^2$$

de donde, haciendo tender $\theta \to 0^+$ se obtiene la desigualdad variacional

$$\phi(\bar{\mathbf{x}}) - \phi(\mathbf{y}) + \frac{1}{\lambda}\langle\mathbf{x}-\bar{\mathbf{x}},\, \mathbf{y}-\bar{\mathbf{x}}\rangle \leq 0, \qquad \forall\, \mathbf{y} \in \mathbb{R}^N \tag{1.38}$$

que necesariamente debe verificar la solución de (1.36). Recíprocamente, si $\bar{\mathbf{x}} \in \mathrm{dom}(\phi)$ verifica (1.38), se tiene que, para cada $\mathbf{y} \in \mathrm{dom}(\phi)$:

$$\begin{aligned}\phi(\bar{\mathbf{x}}) + \frac{1}{2\lambda}|\bar{\mathbf{x}}-\mathbf{x}|^2 &\leq \phi(\mathbf{y}) + \frac{1}{\lambda}\langle\bar{\mathbf{x}}-\mathbf{x},\, \mathbf{y}-\bar{\mathbf{x}}\rangle + \frac{1}{2\lambda}|\bar{\mathbf{x}}-\mathbf{x}|^2\\ &= \phi(\mathbf{y}) + \frac{1}{2\lambda}|\mathbf{y}-\mathbf{x}|^2 - \frac{1}{2\lambda}|\mathbf{y}-\mathbf{x}|^2 + \frac{1}{\lambda}\langle\bar{\mathbf{x}}-\mathbf{x},\, \mathbf{y}-\bar{\mathbf{x}}\rangle + \frac{1}{2\lambda}|\bar{\mathbf{x}}-\mathbf{x}|^2\\ &= \phi(\mathbf{y}) + \frac{1}{2\lambda}|\mathbf{y}-\mathbf{x}|^2 - \frac{1}{2\lambda}|\mathbf{y}-\bar{\mathbf{x}}|^2\end{aligned}$$

de donde es inmediato que $\bar{\mathbf{x}}$ es solución de (1.36).

Una consecuencia inmediata de la caracterización variacional de la solución del problema (1.36) es su unicidad. Así, si suponemos que $\bar{\mathbf{x}}, \bar{\mathbf{z}}$ son soluciones, de (1.38) se obtienen las desigualdades:

$$\phi(\bar{\mathbf{x}}) - \phi(\bar{\mathbf{z}}) + \frac{1}{\lambda}\langle \mathbf{x} - \bar{\mathbf{x}},\, \bar{\mathbf{z}} - \bar{\mathbf{x}}\rangle \leq 0, \qquad \phi(\bar{\mathbf{z}}) - \phi(\bar{\mathbf{x}}) + \frac{1}{\lambda}\langle \mathbf{x} - \bar{\mathbf{z}},\, \bar{\mathbf{x}} - \bar{\mathbf{z}}\rangle \leq 0,$$

que combinadas permiten escribir

$$0 \geq \frac{1}{\lambda}\langle \mathbf{x} - \bar{\mathbf{x}},\, \bar{\mathbf{z}} - \bar{\mathbf{x}}\rangle + \frac{1}{\lambda}\langle \mathbf{x} - \bar{\mathbf{z}},\, \bar{\mathbf{x}} - \bar{\mathbf{z}}\rangle = \frac{1}{\lambda}|\bar{\mathbf{z}} - \bar{\mathbf{x}}|^2.$$

Es decir, $|\bar{\mathbf{z}} - \bar{\mathbf{x}}| = 0 \;\Leftrightarrow\; \bar{\mathbf{x}} = \bar{\mathbf{z}}$. Esta unicidad puede deducirse también de la convexidad estricta de la función objetivo $\mathbf{y} \rightsquigarrow \phi(\mathbf{y}) + \frac{|\mathbf{y}-\mathbf{x}|^2}{2\lambda}$.

Para establecer la existencia de solución es necesario asumir una hipótesis adicional sobre ϕ de tipo topológico.

Definición 1.2 *Se dice que una función $f : \mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$ es* inferiormente semicontinua *(isc) en un punto $\mathbf{x}$ si para cada $\beta < f(\mathbf{x})$, existe $\delta > 0$ de forma que $\beta \leq f(\mathbf{z})$ para cada $\mathbf{z} \in B_\delta(\mathbf{x})$. La función es inferiormente semicontinua en un subconjunto D si lo es en cada uno de sus puntos.*

Es sencillo comprobar que una función f es isc en un punto $\mathbf{x}$ si y solamente si

$$f(\mathbf{x}) \leq \liminf_{\mathbf{z}\to\mathbf{x}} f(\mathbf{z}) = \sup_{\varepsilon>0}\left(\inf_{\mathbf{z}\in B_\varepsilon(\mathbf{x})} f(\mathbf{z})\right). \tag{1.39}$$

Se verifica también un resultado análogo para sucesiones, la denominada *caracterización sucesional de la semicontinuidad inferior*, que establece que una función f es isc en un punto $\mathbf{x}$ si y solamente si para cada sucesión $\mathbf{x}_m \to \mathbf{x}$ se tiene que

$$f(\mathbf{x}) \leq \liminf_{m\to\infty} f(\mathbf{x}_m) = \sup_{M>0}\left(\inf_{m\geq M} f(\mathbf{x}_m)\right). \tag{1.40}$$

En el caso de funciones inferiormente semicontinuas en todo el espacio $\mathbb{R}^N$, se tiene que su epigráfica es un conjunto cerrado. En efecto, si tomamos una sucesión $(\mathbf{x}_m, \alpha_m) \in \mathrm{epi}(f)$, convergente a un punto $(\mathbf{x}, \alpha)$, de la anterior caracterización sucesional, se sigue que

$$f(\mathbf{x}) \leq \liminf_{m\to\infty} f(\mathbf{x}_m) \leq \liminf_{m\to\infty} \alpha_m = \alpha$$

teniendo en cuenta que $f(\mathbf{x}_m) \leq \alpha_m$, para cada m. La desigualdad anterior, $f(\mathbf{x}) \leq \alpha$, implica que $(\mathbf{x}, \alpha) \in \mathrm{epi}(f)$, luego este conjunto es cerrado. Recíprocamente, si $\mathrm{epi}(f) \subset \mathbb{R}^{N+1}$ es cerrado, dado $\beta < f(\mathbf{x})$, es decir, $(\mathbf{x}, \beta) \notin \mathrm{epi}(f)$, debe existir $\eta > 0$ de forma que $(\mathbf{z}, \beta) \notin \mathrm{epi}(f)$, es decir, $\beta < f(\mathbf{z})$, si $\mathbf{z} \in B_\eta(\mathbf{x})$, lo cual implica que f es isc en $\mathbf{x}$. Por tanto,

$$f \text{ es isc} \;\Leftrightarrow\; \mathrm{epi}(f) \subset \mathbb{R}^{N+1} \text{ es cerrado.} \tag{1.41}$$

Combinando la caracterización anterior con la fórmula (1.29) se deduce de forma inmediata el siguiente corolario.

Corolario 1.3 *Sea $\phi_i : \mathbb{R}^N \longrightarrow \mathbb{R} \cup \{+\infty\}$ una familia de funciones inferiormente semicontinuas, $i \in I$. Se tiene que la función*

$$\phi(\mathbf{x}) = \sup_{i \in I} \phi_i(\mathbf{x})$$

es inferiormente semicontinua.

Estamos ya en condiciones de enunciar un resultado de existencia y unicidad de solución para el problema (1.36).

Teorema 1.6 *Sean $\phi : \mathbb{R}^N \longrightarrow \mathbb{R} \cup \{+\infty\}$ una función convexa propia e inferiormente semicontinua y $\lambda > 0$ una constante. Para cada $\mathbf{x} \in \mathbb{R}^N$, el problema (1.36) tiene una única solución que denotamos por $J_\lambda(\mathbf{x})$. Dicha solución queda determinada por la desigualdad variacional*

$$\phi(J_\lambda(\mathbf{x})) - \phi(\mathbf{y}) + \frac{1}{\lambda}\langle \mathbf{x} - J_\lambda(\mathbf{x}),\, \mathbf{y} - J_\lambda(\mathbf{x})\rangle \leq 0, \qquad \forall\ \mathbf{y} \in \mathbb{R}^N \tag{1.42}$$

El campo vectorial $J_\lambda : \mathbb{R}^N \longrightarrow \mathbb{R}^N$ se denomina *resolvente* o *proximal*. De la desigualdad variacional (1.42) se deduce que

$$|J_\lambda(\mathbf{x}) - J_\lambda(\mathbf{z})| \leq |\mathbf{x} - \mathbf{z}|, \qquad \forall\ \mathbf{x}, \mathbf{z} \in \mathbb{R}^N \tag{1.43}$$

es decir, J_λ es 1-Lipschitz. Es también relevante la aplicación que a cada $\mathbf{x} \in \mathbb{R}^N$ le asocia el vector $\frac{1}{\lambda}(\mathbf{x} - J_\lambda(\mathbf{x}))$, denominada *regularización de Yosida*, que es asimismo Lipschitz con constante igual a $1/\lambda$:

$$\left| \frac{1}{\lambda}(\mathbf{x} - J_\lambda(\mathbf{x})) - \frac{1}{\lambda}(\mathbf{z} - J_\lambda(\mathbf{z})) \right| \leq \frac{1}{\lambda}|\mathbf{x} - \mathbf{z}| \tag{1.44}$$

Los anteriores campos vectoriales verifican las desigualdades

$$\left\langle \frac{1}{\lambda}(\mathbf{x} - J_\lambda(\mathbf{x})) - \frac{1}{\lambda}(\mathbf{z} - J_\lambda(\mathbf{z})),\, \mathbf{x} - \mathbf{z} \right\rangle \geq \frac{|\mathbf{x} - J_\lambda(\mathbf{x}) - \mathbf{z} + J_\lambda(\mathbf{z})|^2}{\lambda} \tag{1.45}$$

$$\langle J_\lambda(\mathbf{x}) - J_\lambda(\mathbf{z}),\, \mathbf{x} - \mathbf{z}\rangle \geq |J_\lambda(\mathbf{x}) - J_\lambda(\mathbf{z})|^2 \tag{1.46}$$

de donde se deduce que tanto la regularización de Yosida como la resolvente son campos vectoriales *monótonos*[(1)].

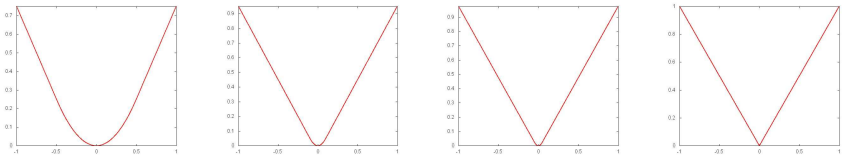
[(1)] La monotonía es la generalización para campos vectoriales de la noción de función creciente.

Ejemplo 1.14 En general no es posible calcular explícitamente la envoltura de Moreau de una función arbitraria. Sin embargo, en el caso particular del valor absoluto, $\phi(x) = |x|$, $x \in \mathbb{R}$, un análisis detallado permite obtener su expresión explícita para cada $\lambda > 0$

$$\phi_\lambda(x) = \begin{cases} -x - \frac{\lambda}{2}, & x \leq -\lambda \\ \frac{x^2}{2\lambda}, & |x| < \lambda \\ x - \frac{\lambda}{2}, & x \geq \lambda \end{cases}$$

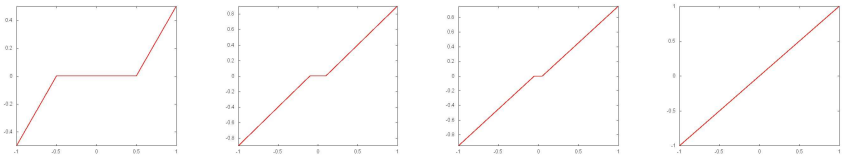
y también la del campo resolvente

$$J_\lambda(x) = \begin{cases} x + \lambda, & x \leq -\lambda \\ 0, & |x| < \lambda \\ x - \lambda, & x \geq \lambda \end{cases}$$



$\lambda = 0.5$ $\lambda = 0.1$ $\lambda = 0.05$ $\lambda = 0$

Figura 1.11: Gráficas de ϕ_λ para $\phi(x) = |x|$



$\lambda = 0.5$ $\lambda = 0.1$ $\lambda = 0.05$ $\lambda = 0$

Figura 1.12: Gráficas de J_λ para $\phi(x) = |x|$

En este ejemplo sencillo, con ayuda de las gráficas anteriores, se observan diversas características que, como veremos posteriormente, se verifican en general. Así

- La envoltura de Moreau ϕ_λ es convexa y derivable para cada $\lambda > 0$, independientemente de la "suavidad" de ϕ.
- $\phi_\lambda \to \phi$ puntualmente cuando $\lambda \to 0$.
- $J_\lambda(x) \to x$ para cada $x \in \mathbb{R}$.

Ejemplo 1.15 Un ejemplo clásico de función convexa es la indicatriz de un convexo $C \subset \mathbb{R}^N$, definida en (1.33). Claramente su envoltura de Moreau es de la forma

$$\inf_{\mathbf{y}\in\mathbb{R}^N}\left(\psi_C(\mathbf{y})+\frac{1}{2\lambda}|\mathbf{x}-\mathbf{y}|^2\right)=\inf_{\mathbf{y}\in C}\left(\frac{1}{2\lambda}|\mathbf{x}-\mathbf{y}|^2\right)=\frac{d_C^2(\mathbf{x})}{2\lambda}$$

mientras que $J_\lambda(\mathbf{x})$ será, para cada $\lambda>0$, la proyección ortogonal $\mathrm{proj}_C(\mathbf{x})$ de $\mathbf{x}$ sobre el convexo C (página 13).

1.2.5 Funciones convexas y continuas

La convexidad de una función es una propiedad de naturaleza geométrica que, no obstante, tiene importantes connotaciones topológicas que analizaremos en este apartado. El resultado principal establece que las funciones convexas son continuas en el interior de su dominio (si éste es no vacío).

Definición 1.3 *Se dice que una función $\phi:\mathbb{R}^N\longrightarrow\mathbb{R}\cup\{+\infty\}$ es* localmente Lipschitz *en el abierto $D\subset\mathbb{R}^N$ si para cada $\mathbf{x}\in D$ se tienen dos constantes $\delta,\alpha(\mathbf{x})>0$ de forma que para cada $\mathbf{y},\mathbf{z}\in B_\delta(\mathbf{x})\subset D$,*

$$|\phi(\mathbf{y})-\phi(\mathbf{z})|\leq\alpha(\mathbf{x})|\mathbf{y}-\mathbf{z}|$$

Teorema 1.7 *Sea $\phi:\mathbb{R}^N\longrightarrow\mathbb{R}\cup\{+\infty\}$ estricta y convexa. Se tiene que ϕ es localmente Lipschitz en el interior de su dominio.*

Nota 1.3 La prueba del teorema anterior (ver [2], [5]) proporciona información adicional sobre la constante de Lipschitz $\alpha(\mathbf{x})$. En efecto, si $\gamma(\mathbf{x})>0$ es una cota de ϕ en un entorno de $\mathbf{x}$, es decir, $|\phi(\mathbf{y})|\leq\gamma(\mathbf{x})$, para cada $\mathbf{y}\in B_\delta(\mathbf{x})$, se tiene que

$$|\phi(\mathbf{y})-\phi(\mathbf{z})|\leq\frac{2\gamma(\mathbf{x})}{\delta/2}|\mathbf{y}-\mathbf{z}| \tag{1.47}$$

si $\mathbf{y},\mathbf{z}\in B_{\delta/2}(\mathbf{x})$,.

Nota 1.4 Aunque en el interior de su dominio una función convexa es localmente Lipschitz y, por tanto, continua, en los puntos de la frontera del dominio puede fallar la continuidad como pone de manifiesto el siguiente ejemplo (ver [15, pág. 83-84] y [16, Example 2.38, pág. 61-62]). Sea el conjunto convexo

$$C=\left\{(x,y)\in\mathbb{R}^2:x+\frac{y^2}{2}\leq 0\right\}$$

y sea σ_C su función soporte, que sabemos que es convexa (ver Ejemplo 1.12). Además, usando el método de los multiplicadores de Lagrange para el cálculo de extremos de funciones, se comprueba que

$$\sigma_C(x,y)=\begin{cases}x^2/(2y), & y>0\\ 0, & (x,y)=(0,0)\\ +\infty, & \text{en otro caso.}\end{cases}$$

Evidentemente la función σ_C es continua en $\operatorname{int}(\operatorname{dom}\sigma_C) = \{(x,y) \in \mathbb{R}^2 : y > 0\}$ (en realidad localmente Lipschitz), pero en el punto $(0,0) \in \operatorname{dom}\sigma_C \cap \partial \operatorname{dom}\sigma_C$ presenta una discontinuidad. En efecto, si nos acercamos al origen de coordenadas siguiendo parábolas de la forma (t, at^2), se tiene que

$$\lim_{t\to 0} \sigma_C(t, at^2) = \lim_{t\to 0} \frac{t^2}{2at^2} = \frac{1}{2a}$$

es decir, no existe el límite de la función cuando $(x,y) \to (0,0)$, con $(x,y) \in \operatorname{dom}\sigma_C$. No obstante σ_C es inferiormente semicontinua en dicho punto, dado que

$$\liminf_{\operatorname{dom}\sigma_C \ni (x,y)\to(0,0)} \sigma_C(x,y) = \sup_{\varepsilon>0} \left(\inf_{(x,y)\in B_\varepsilon(0,0)\cap \operatorname{dom}\sigma_C} \sigma_C(x,y) \right) = 0.$$

1.3 Subdiferenciabilidad

La diferenciabilidad de una función es una propiedad reseñable ya que, entre otras cosas, permite estudiar sus extremos. Las funciones convexas no son necesariamente diferenciables, pero la especial geometría de su epigráfica permite definir hiperplanos soporte en los puntos de su frontera, a partir de los cuales se introducen una especie de "gradientes generalizados", que denominamos *subgradientes*. El conjunto de estos subgradientes (que no son necesariamente únicos en cada punto) se llama subdiferencial y extiende la noción habitual de diferencial en el siguiente sentido: en los puntos de diferenciabilidad la subdiferencial se reduce al gradiente, mientras que es posible asociar un conjunto subdiferencial a puntos donde no existe la diferencial.

Cabe mencionar que la noción de subdiferenciabilidad fue introducida en 1963 de forma independiente por R.T. Rockafellar en su tesis doctoral (a quien se debe la notación $\partial\phi$) y por J.J. Moreau en un artículo en los *Comptes Rendus de l'Académie des Sciences de Paris* (donde se usa por primera vez el término *subgradiente*, del francés *sous-gradient*).

1.3.1 El conjunto subdiferencial

Si ϕ es diferenciable en $\mathbf{x} \in \operatorname{int}(\operatorname{dom}\phi)$, sabemos que $(\mathbf{x}, \phi(\mathbf{x})) \in \partial(\operatorname{epi}\phi)$. Además la frontera de la epigráfica de ϕ es una N-variedad diferenciable descrita por la función $\varphi(\mathbf{y}, \beta) = \phi(\mathbf{y}) - \beta$, y el plano soporte de $\operatorname{epi}\phi$ en dicho punto será de la forma (ver Ejemplo 1.8):

$$\mathcal{H} = \{(\mathbf{y}, \beta) \in \mathbb{R}^{N+1} : \langle \mathbf{y} - \mathbf{x}, \nabla\phi(\mathbf{x})\rangle - (\beta - \phi(\mathbf{x})) = 0\}$$

es decir, $(\nabla\phi(\mathbf{x}), -1)$ es el vector normal al hiperplano soporte de $\operatorname{epi}\phi$ en $(\mathbf{x}, \phi(\mathbf{x}))$. Este hecho motiva la siguiente definición.

Definición 1.4 *Sea* $\phi : \mathbb{R}^N \longrightarrow \mathbb{R} \cup \{+\infty\}$ *una función estricta y convexa. Se dice que* $\mathbf{u} \in \mathbb{R}^N$ *es un* subgradiente *de* ϕ *en* $\mathbf{x} \in \operatorname{dom}\phi$ *si* $(\mathbf{u}, -1)$ *es normal a un hiperplano soporte de la epigráfica en* $(\mathbf{x}, \phi(\mathbf{x}))$*, es decir, si*

$$\langle \mathbf{u}, \mathbf{x} \rangle - \phi(\mathbf{x}) = \sup_{(\mathbf{z},\beta)\in \operatorname{epi}\phi} (\langle \mathbf{u}, \mathbf{z} \rangle - \beta) = \sigma_{\operatorname{epi}\phi}(\mathbf{u}, -1). \tag{1.48}$$

El conjunto de todos los subgradientes se denomina subdiferencial *de* ϕ *en* $\mathbf{x}$ *y se denota por* $\partial\phi(\mathbf{x})$*. Se dice que* ϕ *es* subdiferenciable *en* $\mathbf{x} \in \operatorname{dom}\phi$ *si* $\partial\phi(\mathbf{x}) \neq \emptyset$*.*

Ejemplo 1.16 Sea la función valor absoluto $\phi(x) = |x|$. Si $u \in \partial\phi(0)$ se tiene que

$$0 = \sup_{(x,\beta)\in \operatorname{epi}\phi} (ux - \beta) = \sup_{x\in\mathbb{R}} (ux - |x|) .$$

Si $x > 0$, se tiene que $ux - |x| = x(u-1)$, luego el supremo anterior será igual a cero sobre $\mathbb{R}_+$ si y solamente si $u - 1 \leq 0$. Recíprocamente, si $x < 0$, $ux - |x| = x(u+1)$ y el supremo sobre $\mathbb{R}_-$ será nulo solamente si $u + 1 \geq 0$. Combinando ambos resultados, para que se verifique la identidad anterior, $-1 \leq u \leq 1$, luego

$$\partial\phi(0) = [-1, 1]$$

De este ejemplo sencillo se deducen dos importantes consecuencias:

- Pueden asociarse subgradientes a funciones convexas en puntos donde no son diferenciables.
- En general los gradientes no tienen porqué ser únicos, lo que da sentido a la noción de *conjunto subdiferencial*.

Ejemplo 1.17 Sea la función convexa y estricta

$$\phi(x) = \begin{cases} x^2, & |x| \leq 1 \\ +\infty, & |x| > 1 \end{cases}$$

Se tiene que $u \in \partial\phi(1)$ si y solamente si

$$u - 1 = \sup_{|x|\leq 1,\, \beta \geq x^2} (ux - \beta) = \sup_{|x|\leq 1} \left(ux - x^2\right) \tag{1.49}$$

Analizemos ahora la parábola $h(x) = ux - x^2$, que verifica $h'(x) = u - 2x$, $h''(x) = -2 < 0$. Obviamente $h'(x)$ será decreciente y para cada $-1 < x < 1$

$$h'(1) \leq h'(x) \leq h'(-1) \;\Leftrightarrow\; u - 2 \leq h'(x) \leq u + 2$$

Continuando con el análisis:

- Si $u + 2 \leq 0$, $h'(x) \leq 0$, luego $h(x)$ es decreciente y el máximo se alcanza en $x = -1$. Sustituyendo en (1.49) se tiene que $u - 1 = h'(-1) = -u - 1 \;\Leftrightarrow\; u = 0$, lo cual es absurdo.

- Si $u+2 \geq 0$, $h'(x) \geq 0$, y la función será creciente, por lo que el máximo se alcanza en $x=1$. Sustituyendo en (1.49) vemos que la identidad se verifica trivialmente, es decir, $[2,+\infty[\subset \partial\phi(1)$.
- Si $-2 < u < 2$, el polinomio $h(x)$ tiene un máximo en el punto $x = u/2 \in]-1,1[$, luego

$$u - 1 = h(u/2) = \frac{u^2}{4} \Leftrightarrow 0 = u^2 - 4u + 4 = (u-2)^2$$

  lo cual es absurdo.

Recapitulando, $\partial\phi(1) = [2,+\infty[$.

Ejemplo 1.18 ([15], p. 215) Sea la función convexa

$$f(x) = \begin{cases} -(1-x^2)^{1/2}, & |x| \leq 1 \\ +\infty, & |x| > 1 \end{cases}$$

Usando la definición de subgradiente, $u \in \partial f(1)$ si y solamente si

$$-1 = \sup_{|x|\leq 1} ux + (1+x^2)^{1/2}$$

Pero, si $u \geq 0$, el supremo de la derecha será obviamente positivo, luego $\partial f(1) \subset]-\infty,0[$. Sea la función $h(x) = ux + (1-x^2)^{1/2}$, cuya derivada es de la forma

$$h'(x) = u - x(1-x^2)^{1/2} \leq 0$$

si $u < 0$, es decir, $h(x)$ es decreciente y su máximo se alcanzará en $x = -1$. Por tanto, si $u \in \partial f(-1)$, $-1 = h(-1) = -u$, lo cual es absurdo. Así, pues, $\partial f(1) = \emptyset$.

Ejemplo 1.19 La función indicatriz de un convexo, $C \subset \mathbb{R}^N$, es convexa, además $\mathbf{u} \in \partial\psi_C(\mathbf{x})$ si

$$\langle \mathbf{u}, \mathbf{x} \rangle = \sup_{(\mathbf{z},\beta)\in \operatorname{epi}\psi_C} (\langle \mathbf{u}, \mathbf{z} \rangle - \beta) = \sup_{\mathbf{z}\in C} \langle \mathbf{u}, \mathbf{z} \rangle = \sigma_C(\mathbf{u})$$

es decir, la subdiferencial de ψ_C en $\mathbf{x} \in \partial C$ es el conjunto de los vectores normales a los hiperplanos que soportan al conjunto en dicho punto. Por el contrario, si $\mathbf{x} \in \operatorname{int} C$, para cada $\mathbf{u} \in \mathbb{R}^N \setminus \{0\}$, si $t > 0$ es lo suficientemente pequeño, $\mathbf{x} + t\mathbf{u} \in C$, de donde

$$\sigma_C(\mathbf{u}) \geq \langle \mathbf{u}, \mathbf{x} + t\mathbf{u} \rangle = \langle \mathbf{u}, \mathbf{x} \rangle + t|\mathbf{u}|^2 > \langle \mathbf{u}, \mathbf{x} \rangle.$$

Es decir, $\partial\psi_C(\mathbf{x}) = \{\mathbf{0}\}$, si $\mathbf{x} \in \operatorname{int} C$.

Obviamente, si $\mathbf{u}, \mathbf{v} \in \partial\phi(\mathbf{x})$ y $0 < \lambda < 1$, se tiene que

$$\begin{aligned} \langle (1-\lambda)\mathbf{u} + \lambda\mathbf{v}, \mathbf{x} \rangle &= (1-\lambda)\langle \mathbf{u}, \mathbf{x} \rangle + \lambda \langle \mathbf{v}, \mathbf{x} \rangle \\ &= \phi(\mathbf{x}) + (1-\lambda)\sigma_{\operatorname{epi}\phi}(\mathbf{u}, -1) + \lambda\sigma_{\operatorname{epi}\phi}(\mathbf{v}, -1) \\ &\geq \phi(\mathbf{x}) + \sigma_{\operatorname{epi}\phi}((1-\lambda)\mathbf{u} + \lambda\mathbf{v}) \end{aligned}$$

usando la convexidad de la función soporte. De aquí es evidente la inclusión $(1-\lambda)\mathbf{u}+\lambda\mathbf{v} \in \partial\phi(\mathbf{x})$, es decir, el conjunto subdiferencial es convexo. Además, si $\mathbf{u}_m \to \mathbf{u}$, con $\mathbf{u}_m \in \partial\phi(\mathbf{x})$, de la definición de subgradiente se tiene que para cada $(\mathbf{z},\beta) \in \operatorname{epi}\phi$,

$$\langle \mathbf{u}_m, \mathbf{x}\rangle - \phi(\mathbf{x}) \geq \langle \mathbf{u}_m, \mathbf{z}\rangle - \beta$$

y tomando límite cuando $m \to \infty$, podemos escribir la relación $\langle \mathbf{u}, \mathbf{x}\rangle - \phi(\mathbf{x}) \geq \langle \mathbf{u}, \mathbf{z}\rangle - \beta$, que equivale a $\mathbf{u} \in \partial\phi(\mathbf{x})$. Esto significa que el conjunto subdiferencial es cerrado. Estas dos propiedades se recopilan en la siguiente proposición.

Proposición 1.11 *Sea $\phi : \mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$ una función convexa y estricta. Dado $\mathbf{x} \in \operatorname{dom}\phi$, de forma que $\partial\phi(\mathbf{x}) \neq \emptyset$, se tiene que el conjunto subdiferencial es convexo y cerrado.*

Concluimos la sección estableciendo la subdiferenciablidad de las funciones convexas en el interior de su dominio.

Teorema 1.8 (Subdiferenciabilidad) *Sea $\phi : \mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$ una función convexa. Para cada $\mathbf{x} \in \operatorname{int}(\operatorname{dom}\phi)$, se tiene que $\partial\phi(\mathbf{x}) \neq \emptyset$.*

Nota 1.5 Como pone de manifiesto el Ejemplo 1.18, el conjunto subdiferencial puede ser vacío en puntos de la frontera del dominio de una función convexa.

1.3.2 Derivadas direccionales

Dada una función $\phi : \mathbb{R}^N \longrightarrow \mathbb{R}\cup\{\infty\}$, se define su derivada direccional (también llamada variación de Gâteaux) en el punto $\mathbf{x} \in \operatorname{dom}\phi$ en la dirección $\mathbf{u} \in \mathbb{R}^N \setminus \{\mathbf{0}\}$ como el valor del límite (cuando existe):

$$\lim_{h\to 0^+} \frac{\phi(\mathbf{x}+h\mathbf{u}) - \phi(\mathbf{x})}{h}$$

La denotaremos por $\delta\phi(\mathbf{x};\mathbf{u})$. Cuando ϕ es diferenciable en $\mathbf{x} \in \operatorname{int}(\operatorname{dom}\phi)$ es evidente que $\delta\phi(\mathbf{x};\mathbf{u}) = \langle \nabla\phi(\mathbf{x}), \mathbf{u}\rangle$.

En el caso de las funciones convexas siempre es posible calcular derivadas direccionales en los puntos del interior del dominio en cualquier dirección. Además, las derivadas direccionales permiten caracterizar la subdiferencial.

Lema 1.1 *Sea $\phi : \mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$ convexa y estricta. Para cada $\mathbf{x} \in \operatorname{int}(\operatorname{dom}\phi)$ se tiene que*

(i) Existe $\delta\phi(\mathbf{x};\mathbf{u})$ para cada $\mathbf{u} \in \mathbb{R}^N \setminus \{\mathbf{0}\}$.

(ii) La aplicación $\mathbf{u} \rightsquigarrow \delta\phi(\mathbf{x};\mathbf{u})$ es

- *Positivamente homogénea: $\delta\phi(\mathbf{x};\mu\mathbf{u}) = \mu\delta\phi(\mathbf{x};\mathbf{u})$, si $\mu > 0$.*
- *Subaditiva: $\delta\phi(\mathbf{x};\mathbf{u}+\mathbf{v}) \leq \delta\phi(\mathbf{x};\mathbf{u}) + \delta\phi(\mathbf{x};\mathbf{v})$.*

(iii) La aplicación anterior es continua.

Nota 1.6 De la homogeneidad de las variaciones de Gâteaux se desprende que el valor de las derivadas direccionales queda determinado por $\delta\phi(\mathbf{x};\mathbf{u})$ para $|\mathbf{u}| = 1$. Por ejemplo, si $\phi(x) = |x|$ es la función valor absoluto, se tiene que

$$\delta\phi(0;1) = \lim_{t\to 0^+} \frac{|t|}{t} = 1, \qquad \delta\phi(0;-1) = \lim_{t\to 0^+} \frac{|-t|}{t} = 1$$

de donde es inmediato comprobar que $\delta\phi(\mathbf{x};\mathbf{u}) = |\mathbf{u}|$. Este ejemplo pone además de manifiesto que la aplicación variación de Gâteaux no es lineal en general.

Proposición 1.12 *Sea $\phi : \mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$ una función convexa y estricta. Dado $\mathbf{x} \in \operatorname{dom}\phi$, se tiene que*

$$\partial\phi(\mathbf{x}) = \left\{\mathbf{u} \in \mathbb{R}^N : \langle \mathbf{u}, \mathbf{v}\rangle \leq \delta\phi(\mathbf{x};\mathbf{v}),\ \forall\ \mathbf{v} \in \mathbb{R}^N\right\}. \tag{1.50}$$

Corolario 1.4 *Sea $\phi : \mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$ convexa y estricta. Para cada $\mathbf{x} \in \operatorname{dom}\phi$ se tiene que $\mathbf{u} \in \partial\phi(\mathbf{x})$ si y solamente si para cada $\mathbf{y} \in \mathbb{R}^N$,*

$$\langle \mathbf{u}, \mathbf{y}-\mathbf{x}\rangle \leq \phi(\mathbf{y}) - \phi(\mathbf{x})$$

Corolario 1.5 *Sean $\phi : \mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$ convexa y estricta, $\mathbf{x} \in \operatorname{dom}\phi$. Se tiene que:*

(i) Si $\partial\phi(\mathbf{x}) \neq \emptyset$, entonces ϕ es inferiormente semicontinua en $\mathbf{x}$.

*(ii) $\mathbf{0} \in \partial\phi(\mathbf{x}) \Leftrightarrow \mathbf{x}$ es un mínimo de ϕ (*Regla de Fermat*).*

Corolario 1.6 *Sea $\phi : \mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$ convexa y estricta. Si $\mathbf{x} \in \operatorname{int}(\operatorname{dom}\phi)$, el conjunto $\partial\phi(\mathbf{x})$ está acotado. En concreto, si $\alpha(\mathbf{x})$ es la constante de Lipschitz de ϕ en un entorno de $\mathbf{x}$, se verifica:*

$$\sup_{\mathbf{u}\in\partial\phi(\mathbf{x})} |\mathbf{u}| \leq \alpha(\mathbf{x}) \tag{1.51}$$

Es importante resaltar que la cota de la subdiferencial tiene carácter local. En efecto, si ϕ es $\alpha(\mathbf{x})$-Lipschitz en $B_\varepsilon(\mathbf{x}) \subset \operatorname{int}(\operatorname{dom}\phi)$, se tiene que

$$\sup_{\mathbf{z}\in B_\varepsilon(\mathbf{x})} \left(\sup_{\mathbf{u}\in\partial\phi(\mathbf{z})} |\mathbf{u}|\right) \leq \alpha(\mathbf{x}) \ \Leftrightarrow\ \partial\phi(\mathbf{z}) \subset \alpha(\mathbf{x})\mathcal{B},\ \ \forall\ \mathbf{z} \in B_\varepsilon(\mathbf{x}) \tag{1.52}$$

Es más, si $K \subset \operatorname{int}(\operatorname{dom}\phi)$ es un conjunto compacto, existirán $\mathbf{x}_1, \ldots, \mathbf{x}_m$ en K, $\varepsilon_i > 0$, $1 \leq i \leq m$, de forma que $\overline{B}_{\varepsilon_i}(\mathbf{x}_i) \subset \operatorname{int}(\operatorname{dom}\phi)$ y

$$K \subset \bigcup_{i=1}^{m} B_{\varepsilon_i/2}(\mathbf{x}_i).$$

Por otra parte, al ser las bolas cerradas conjuntos compactos, de la continuidad de ϕ se tiene $\gamma(\mathbf{x}_i) > 0$ tal que $|\phi(\mathbf{x})| \leq \gamma(\mathbf{x}_i)$, para cada $\mathbf{x} \in B_{\varepsilon_i}(\mathbf{x}_i)$, de donde, usando la Nota 1.3, (1.52) nos permite escribir, para cada $\mathbf{x} \in B_{\varepsilon_i/2}(\mathbf{x}_i)$,

$$\partial\phi(\mathbf{x}) \subset \frac{2\gamma(\mathbf{x}_i)}{\varepsilon_i/2}\mathcal{B}.$$

Finalmente, tomando $\gamma = \max_{1\leq i\leq m}\gamma(\mathbf{x}_i)$, $\varepsilon = \min_{1\leq i\leq m}\varepsilon_i > 0$ es evidente que

$$\sup_{\mathbf{x}\in K}\left(\sup_{\mathbf{u}\in\partial\phi(\mathbf{x})}|\mathbf{u}|\right) \leq \frac{2\gamma}{\varepsilon/2} \tag{1.53}$$

La gráfica o grafo de la subdiferencial se define como el conjunto

$$\mathrm{Gr}(\partial\phi) = \left\{(\mathbf{x},\mathbf{u}) \in \mathbb{R}^{2N} : \mathbf{u} \in \partial\phi(\mathbf{x})\right\}. \tag{1.54}$$

Si la función ϕ es inferiormente semicontinua, el conjunto anterior es cerrado.

Corolario 1.7 *Sea $\phi : \mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$ estricta, convexa e inferiormente semicontinua. Se tiene que la gráfica de la subdiferencial es un conjunto cerrado. En particular, si $(\mathbf{x}_m,\mathbf{u}_m) \to (\mathbf{x},\mathbf{u})$, con $\mathbf{u}_m \in \partial\phi(\mathbf{x}_m)$, se tiene que $\mathbf{u} \in \partial\phi(\mathbf{x})$.*

Para concluir el apartado veamos que, tal como se indicaba al inicio de la sección, la noción de subdiferencial extiende el concepto clásico de gradiente. La demostración de esta afirmación se basa en la caracterización de las variaciones de Gâteaux/derivadas direccionales como función soporte del conjunto subdiferencial.

Lema 1.2 *Sea $\phi : \mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$ estricta y convexa. Si $\mathbf{x} \in \mathrm{int}\,(\mathrm{dom}\,\phi)$ se verifica la identidad*

$$\delta\phi(\mathbf{x};\mathbf{v}) = \sigma_{\partial\phi(\mathbf{x})}(\mathbf{v}), \quad \forall\ \mathbf{v} \in \mathbb{R}^N. \tag{1.55}$$

Teorema 1.9 *Sean $\phi : \mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$ estricta y convexa, $\mathbf{x} \in \mathrm{int}\,(\mathrm{dom}\,\phi)$. Se tiene que ϕ es diferenciable en $\mathbf{x}$ si y solamente si $\partial\phi(\mathbf{x}) = \{\nabla\phi(\mathbf{x})\}$.*

1.3.3 Envoltura de Moreau II

Volvamos ahora a la envoltura de Moreau, a la que ya dedicamos el apartado 1.2.4. En primer lugar, de la desigualdad (1.42) y el Corolario 1.4 es evidente que la regularización de Yosida en cada punto está contenida en el conjunto subdiferencial de ϕ en su resolvente, es decir,

$$\frac{1}{\lambda}\left(\mathbf{x} - J_\lambda(\mathbf{x})\right) \in \partial\phi(J_\lambda(\mathbf{x})). \tag{1.56}$$

Teniendo en cuenta este hecho, si $\mathbf{x} \in \operatorname{int}(\operatorname{dom}\phi)$, se considera el subgradiente de menor norma, $\partial\phi^0(\mathbf{x}) = \operatorname{proj}_{\partial\phi(\mathbf{x})}(\mathbf{0})$, que verifica la fórmula:

$$\left|\frac{1}{\lambda}\left(\mathbf{x} - J_\lambda(\mathbf{x})\right) - \partial\phi^0(\mathbf{x})\right|^2 \leq |\partial\phi^0(\mathbf{x})|^2 - \left|\frac{1}{\lambda}\left(\mathbf{x} - J_\lambda(\mathbf{x})\right)\right|^2. \tag{1.57}$$

De aquí, usando (1.51), se llega a

$$|\mathbf{x} - J_\lambda(\mathbf{x})| \leq \lambda|\partial\phi^0(\mathbf{x})| \leq \lambda\alpha(\mathbf{x}) \underset{\lambda\to 0^+}{\longrightarrow} 0 \tag{1.58}$$

es decir, la resolvente converge a la identidad puntualmente en el interior del dominio de ϕ. Es más, si $K \subset \operatorname{int}(\operatorname{dom}\phi)$ es un compacto, de (1.53), existirá $\alpha > 0$ de forma que

$$\sup_{\mathbf{x}\in K} |\mathbf{x} - J_\lambda(\mathbf{x})| \leq \lambda \sup_{\mathbf{x}\in K} |\partial\phi^0(\mathbf{x})| \leq \lambda\alpha \underset{\lambda\to 0^+}{\longrightarrow} 0 \tag{1.59}$$

lo que significa que la convergencia de la resolvente hacia la identidad es uniforme sobre los compactos.

Por otra parte, es inmediato comprobar que la envoltura de Moreau de una función estricta, convexa e inferiormente semicontinua, definida por (1.35), es una función convexa, cuya subdiferencial en cada punto contiene a la regularización de Yosida

$$\frac{1}{\lambda}\left(\mathbf{x} - J_\lambda(\mathbf{x})\right) \in \partial\phi_\lambda(\mathbf{x}). \tag{1.60}$$

Es más, ϕ_λ es diferenciable con

$$\nabla\phi_\lambda(\mathbf{x}) = \frac{1}{\lambda}\left(\mathbf{x} - J_\lambda(\mathbf{x})\right). \tag{1.61}$$

Veamos ahora hacia qué converge la envoltura de Moreau cuando $\lambda \to 0^+$. Claramente $\phi_\lambda(\mathbf{x}) \leq \phi(\mathbf{x})$, luego

$$\limsup_{\lambda\to 0^+} \phi_\lambda(\mathbf{x}) \leq \phi(\mathbf{x}). \tag{1.62}$$

Además, de la definición de resolvente, $\phi_\lambda(\mathbf{x}) = \phi(J_\lambda(\mathbf{x})) + \frac{1}{2\lambda}|J_\lambda(\mathbf{x}) - \mathbf{x}|^2$, por lo que al ser ϕ inferiormente semicontinua podemos escribir[(2)]

$$\begin{aligned} \phi(\mathbf{x}) &\leq \liminf_{\lambda\to 0^+} \phi(J_\lambda(\mathbf{x})) \leq \liminf_{\lambda\to 0^+} \phi(J_\lambda(\mathbf{x})) + \frac{1}{2\lambda}|J_\lambda(\mathbf{x}) - \mathbf{x}|^2 \\ &= \liminf_{\lambda\to 0^+} \phi_\lambda(\mathbf{x}) \end{aligned} \tag{1.63}$$

[(2)] En realidad

$$\liminf_{\lambda\to 0^+} \phi(J_\lambda(\mathbf{x})) = \liminf_{\lambda\to 0^+} \phi(J_\lambda(\mathbf{x})) + \frac{1}{2\lambda}|J_\lambda(\mathbf{x}) - \mathbf{x}|^2$$

teniendo en cuenta que $\frac{1}{2\lambda}|J_\lambda(\mathbf{x}) - \mathbf{x}|^2 \leq \frac{\lambda^2\alpha(\mathbf{x})^2}{2\lambda} = \frac{\lambda\alpha(\mathbf{x})^2}{2} \to 0$, cuando $\lambda \to 0^+$.

De las desigualdades (1.62)-(1.63) se tiene que existe el límite puntual de la envoltura de Moreau y

$$\lim_{\lambda\to 0^+} \phi_\lambda(\mathbf{x}) = \phi(\mathbf{x}) \tag{1.64}$$

También existe el límite puntual de la regularización de Yosida,

$$\hat{\mathbf{y}}(\mathbf{x}) = \lim_{\lambda\to 0^+} \frac{1}{\lambda}\left(\mathbf{x} - J_\lambda(\mathbf{x})\right) \tag{1.65}$$

para cada $\mathbf{x} \in \operatorname{int}(\operatorname{dom}\phi)$. Al ser el grafo de la subdiferencial un conjunto cerrado, Corolario 1.7, de las relaciones (1.56), (1.58), (1.65) se tiene que

$$\hat{\mathbf{y}}(\mathbf{x}) \in \partial\phi(\mathbf{x}),$$

con $|\hat{\mathbf{y}}(\mathbf{x})| \leq |\partial\phi^0(\mathbf{x})|$, como consecuencia de la desigualdad (1.57). Esto implica $\hat{\mathbf{y}}(\mathbf{x}) = \partial\phi^0(\mathbf{x})$ por unicidad del subgradiente de mínima norma (que es la proyección ortogonal sobre el convexo $\partial\phi(\mathbf{x})$ del vector nulo). En resumen, para cada $\mathbf{x} \in \operatorname{int}(\operatorname{dom}\phi)$, se tiene que

$$\lim_{\lambda\to 0^+} \frac{1}{\lambda}\left(\mathbf{x} - J_\lambda(\mathbf{x})\right) = \partial\phi^0(\mathbf{x}). \tag{1.66}$$

1.4 Inclusiones diferenciales de tipo subdiferencial

Las inclusiones de tipo subdiferencial constituyen una extensión de la noción usual de sistema de ecuaciones diferenciales de tipo gradiente

$$-\dot{\mathbf{x}}(t) = \nabla\phi(\mathbf{x}(t)) \tag{1.67}$$

que permiten manejar potenciales convexos e inferiormente semicontinuos no necesariamente diferenciables.

El estudio de esta clase de problemas se inició alrededor de la década de los 70 del pasado siglo, destacando las aportaciones de H. Brézis y J.J. Moreau. Para más detalles acerca del origen y del desarrollo histórico de la teoría de las inclusiones diferenciales (o ecuaciones de evolución) de tipo subdiferencial nos remitimos a [3] y, especialmente, a [10].

A lo largo de la sección $\phi : \mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$ será una función estricta, convexa e inferiormente semicontinua, que también puede depender de una variable real t (asociada al tiempo), $\phi : [0,T[\times\mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$, $(0 < T \leq +\infty)$ en un sentido que se precisará más adelante.

1.4.1 Tipos de problemas

La clase más simple de ecuaciones de tipo subdiferencial son las de la forma

$$-\dot{\mathbf{x}}(t) \in \partial\phi(\mathbf{x}(t)). \tag{1.68}$$

La referencia clásica es [4] donde se estudian en el marco general de los operadores maximales monótonos definidos en espacios de dimensión no necesariamente finita. Otro texto donde es trata ampliamente este problema es [3].

En el caso en que el potencial varía con el tiempo, se tienen las ecuaciones

$$-\dot{\mathbf{x}}(t) \in \partial\phi(t, \mathbf{x}(t)), \tag{1.69}$$

donde $\partial\phi(t, \mathbf{x})$ denota la subdiferencial respecto de la variable $\mathbf{x}$ para un valor de t fijo. Esta clase de ecuaciones fue estudiada por primera vez por Moreau en el caso particular $\phi(t, \mathbf{x}) = \sigma_{C(t)}(\mathbf{x})$, con $C(t)$ una familia de conjuntos de $\mathbb{R}^N$ variando con el tiempo. Consideraremos también el caso en que existe un término fuente dado por un campo vectorial $\mathbf{F} : [0, T[\times \mathbb{R}^N \longrightarrow \mathbb{R}^N$, que adicionalmente puede depender también del tiempo, lo que nos da la ecuación

$$-\dot{\mathbf{x}}(t) \in \partial\phi(t, \mathbf{x}(t)) - \mathbf{F}(t, \mathbf{x}(t)). \tag{1.70}$$

Las inclusiones diferenciales asociadas a potenciales variables, con y sin término fuente, se tratan con detalle en el segundo capítulo de [10].

Las denominadas *ecuaciones bipotenciales* son aquellas en las que aparecen involucrados dos subdiferenciales. En esta memoria consideraremos la forma general

$$-\dot{\mathbf{x}}(t) \in \partial\phi(t, \mathbf{x}(t)) + \beta(t, \mathbf{x}(t))\partial\varphi(t, \mathbf{x}(t)) \tag{1.71}$$

donde $\beta : [0, T[\times \mathbb{R}^N \longrightarrow \mathbb{R}$ es una función, aunque lo más habitual es suponer que los potenciales son estacionarios, con lo que el problema toma la forma

$$-\dot{\mathbf{x}}(t) \in \partial\phi(\mathbf{x}(t)) + \beta(t)\partial\varphi(\mathbf{x}(t)) \tag{1.72}$$

Cabe decir que bajo ciertas hipótesis, por ejemplo,

- Ambos potenciales son funciones independientes del tiempo, estrictas, convexas e inferiormente semicontinuas, de forma que

$$\mathbf{0} \in \operatorname{int}(\operatorname{dom}\phi - \operatorname{dom}\varphi). \tag{1.73}$$

- β toma valores no negativos y solamente depende de t.

se tiene la identidad

$$\partial\phi(\mathbf{x}) + \beta(t)\partial\varphi(\mathbf{x}) = \partial\left(\phi + \beta(t)\varphi\right)(\mathbf{x}), \tag{1.74}$$

con lo que (1.72) no es mas que un caso particular de (1.69), a pesar de lo cual sigue teniendo interés (ver p.e. [1]). Sin embargo (1.74) no se da en general, lo que obliga a tratar este problema de forma independiente. Tienen particular interés el problema denominado de *subdiferenciales opuestas* o *diferencia de subdiferenciales* en que $\beta = -1$, que se analiza detalladamente en el apartado II.4 de [10] y el caso $\varphi(t, \mathbf{x}) = d_{C(t)}(\mathbf{x})$ asociado a problemas de viabilidad (ver p.e. [7]).

1.4.2 Existencia de solución

Definición 1.5 *Se denomina* solución en sentido fuerte (strong solution) o de Brézis *de (1.69) en un intervalo* $[0,T[$, $(0 < T \leq +\infty)$ *a una función absolutamente continua*[3] $\mathbf{x} : [0,T[\longrightarrow \mathbb{R}^N$ *de forma que, para casi todo* $0 < t < T$*:*

(i) $\partial\phi(t,\mathbf{x}(t)) \neq \emptyset$.

(ii) $-\dot{\mathbf{x}}(t) \in \partial\phi(t,\mathbf{x}(t))$.

Si el potencial no depende del tiempo, problema (1.68), la noción de solución es análoga, mientras que para el caso perturbado (1.70), simplemente se sustituye la condición (ii) por

(ii)* $-\dot{\mathbf{x}}(t) + \mathbf{F}(t,\mathbf{x}(t)) \in \partial\phi(t,\mathbf{x}(t))$.

Finalmente, una función absolutamente continua $\mathbf{x} : [0,T[\longrightarrow \mathbb{R}^N$ *es solución de la ecuación bipotencial (1.72) si para casi todo* $0 < t < T$*,*

(i)** $\partial\phi(t,\mathbf{x}(t)) \neq \emptyset$, $\partial\varphi(t,\mathbf{x}(t)) \neq \emptyset$

(ii)** *Existen funciones* $\boldsymbol{f},\boldsymbol{g} : [0,T[\longrightarrow \mathbb{R}^N$ *integrables de forma que, para casi todo* $0 < t < T$, $\boldsymbol{f}(t) \in \partial\phi(t,\mathbf{x}(t))$, $\boldsymbol{g}(t) \in \partial\varphi(t,\mathbf{x}(t))$.

(iii)** *Para casi todo* $0 < t < T$, $-\dot{\mathbf{x}}(t) = \boldsymbol{f}(t) + \beta(t,\mathbf{x}(t))\boldsymbol{g}(t)$.

El problema de Cauchy o de condición inicial para las ecuaciones anteriores está bien puesto en el sentido de Hadamard, es decir, para cada condición inicial admisible $\mathbf{x}_0$ existe una única solución de la ecuación correspondiente, de forma que

$$\mathbf{x}(0) = \mathbf{x}_0. \tag{1.75}$$

[3] Un conjunto $A \subset \mathbb{R}$ se dice que *tiene medida cero* si dado $\varepsilon > 0$ arbitrario, existen sucesiones $a_m < b_m$, $m \geq 1$, de forma que

(1) $A \subset \bigcup_{m=1}^{\infty} [a_m, b_m]$.

(2) $\sum_{m=1}^{\infty} (b_m - a_m) \leq \varepsilon$

Los conjuntos finitos y las sucesiones son ejemplos de este tipo de conjuntos. Una cierta propiedad se dice que se verifica *casi por todas partes* (abreviadamente c.t.p.) en un intervalo $I \subset \mathbb{R}$ si solamente deja de verificarse en un subconjunto de puntos de medida nula. Una función $f : I \subseteq \mathbb{R} \longrightarrow \mathbb{R}^N$ se dice que es *absolutamente continua* si

(1) Es derivable en todo el intervalo excepto a lo sumo en un subconjunto de medida nula, es decir, se tiene $A \subset [a,b]$ de medida nula, de forma que para cada $t \notin A$ existe el límite

$$\lim_{h \to 0} \frac{f(t+h) - f(t)}{h} = f'(t)$$

(2) La función f', definida c.t.p. en el intervalo I es integrable y, para cada $s,t \in I$, $s < t$, se verifica la identidad

$$f(t) - f(s) = \int_s^t f'(\tau)d\tau$$

Además la solución depende de forma continua de la condición inicial. Analicemos ahora con más detalle este resultado para cada problema particular.

▶ Ecuación (1.68): potencial independiente del tiempo. Sea una función $\phi : \mathbb{R}^N \longrightarrow \mathbb{R} \cup \{+\infty\}$ estricta, convexa e inferiormente semicontinua.

Teorema 1.10 (Dependencia continua y unicidad) *Si $\mathbf{x}(\cdot)$, $\mathbf{y}(\cdot)$ son soluciones de (1.68) para las condiciones iniciales $\mathbf{x}_0$, $\mathbf{y}_0$, respectivamente, se tiene que, para cada $t > 0$,*

$$|\mathbf{x}(t) - \mathbf{y}(t)| \leq |\mathbf{x}_0 - \mathbf{y}_0| \tag{1.76}$$

Obviamente la estimación (1.76) implica que las soluciones dependen de forma continua de las condiciones iniciales y que no pueden haber dos soluciones distintas de la misma ecuación verificando la misma condición inicial (unicidad).

Teorema 1.11 (Existencia) *Para cada estado inicial admisible $\mathbf{x}_0 \in \overline{\operatorname{dom}\phi}$, existe una única solución de (1.68)+(1.75), $\mathbf{x}(\cdot)$, definida en $[0, +\infty[$. Además:*

(i) Para todo $t \geq 0$, se tiene que

$$\dot{\mathbf{x}}(t) = \lim_{h \to 0^+} \frac{\mathbf{x}(t+h) - \mathbf{x}(t)}{h}$$

y la función $t \rightsquigarrow \dot{\mathbf{x}}(t)$ es continua por la derecha.

(ii) Para casi todo $t > 0$, $-\dot{\mathbf{x}}(t) = \partial\phi^0(\mathbf{x}(t))$.

(iii) La función $t \rightsquigarrow |\dot{\mathbf{x}}(t)|$ es decreciente.

(iv) La función $t \rightsquigarrow \phi(\mathbf{x}(t))$ es convexa y para casi todo $t > 0$ se verifica la identidad

$$\frac{d}{dt}\left(\phi(\mathbf{x}(t))\right) + |\dot{\mathbf{x}}(t)|^2 = 0, \tag{1.77}$$

de donde se deduce que las soluciones de (1.68) proporcionan trayectorias de descenso de ϕ.

(v) Si ϕ admite mínimo, $\operatorname{argmin}\phi \neq \emptyset$, se verifica:

- *Convergencia asintótica:*

$$\lim_{t \to +\infty} \mathbf{x}(t) \in \operatorname{argmin}\phi \tag{1.78}$$

- *Convergencia ergódica:*

$$\lim_{t \to +\infty} \frac{1}{t} \int_0^t \mathbf{x}(s)ds \in \operatorname{argmin}\phi \tag{1.79}$$

De las propiedades de la envoltura o regularización de Moreau (Apartados 1.2.4, 1.3.3) y el Teorema de Picard-Lindelöf para ecuaciones diferenciales ordinarias, se tiene que, para cada $\lambda > 0$ existe una única solución del problema de Cauchy

$$\begin{cases} -\dot{\mathbf{x}}_\lambda(t) = \nabla\phi_\lambda(\mathbf{x}_\lambda(t)), \\ \mathbf{x}_\lambda(0) = \mathbf{x}_\lambda, \end{cases} \tag{1.80}$$

definida en el intervalo $[0, +\infty[$, con $\partial\phi(\mathbf{x}_\lambda) \neq \emptyset$ y $\mathbf{x}_\lambda \to \mathbf{x}_0$. Además la familia de las soluciones $\mathbf{x}_\lambda(\cdot)$ converge hacia una función $\mathbf{x}(\cdot)$ cuando $\lambda \to 0^+$, que es la solución de (1.68)+(1.75). Esta es esencialmente la prueba del Teorema 1.11, para los detalles nos remitimos a [3] y [4].

► Ecuación (1.69): potencial variable. Sea $\phi : [0, T[\times \mathbb{R}^N \longrightarrow \mathbb{R} \cup \{+\infty\}$ ($T \in \mathbb{R}$) estricta de forma que

(ϕ1) Para cada $0 < t < T$, $\phi(t, \cdot)$ es convexa e inferiomente semicontinua.

(ϕ2) Para cada $r > 0$ se tienen $\beta_r > 0$, g_r, h_r de clase C^1 de forma que para cada $0 < t < T$, $\mathbf{x} \in \operatorname{dom}\phi(t, \cdot) \cap B_r(\mathbf{0})$, si $t < s < T$, existe $\widehat{\mathbf{x}} \in \mathbb{R}^N$ con

- $\partial\phi(s, \widehat{\mathbf{x}}) \neq \emptyset$
- $|\widehat{\mathbf{x}} - \mathbf{x}| \leq |g_r(s) - g_r(t)| \, (\phi(t, \mathbf{x}) + \beta_r)^{1/2}$
- $\phi(s, \widehat{\mathbf{x}}) \leq \phi(t, \mathbf{x}) + |h_r(s) - h_r(t)| \, (\phi(t, \mathbf{x}) + \beta_r)$

Si $\mathbf{x}(\cdot)$, $\mathbf{y}(\cdot)$ son soluciones de (1.69) para las condiciones iniciales $\mathbf{x}_0$, $\mathbf{y}_0$, respectivamente, se verifica la desigualdad (1.76), lo que proporciona la dependencia continua de las condiciones iniciales y la unicidad para el problema de Cauchy (1.69)+(1.75). En cuanto a la existencia, las condiciones (ϕ1)-(ϕ2) garantizan que la función $\nabla\phi_\lambda(t, \mathbf{x}) = \frac{1}{\lambda}(\mathbf{x} - J_\lambda(t, \mathbf{x}))$ es continua respecto de la variable t y Lipschitz respecto de $\mathbf{x}$, por lo que los problemas

$$\begin{cases} -\mathbf{x}_\lambda(t) = \nabla\phi_\lambda(t, \mathbf{x}_\lambda(t)), \\ \mathbf{x}_\lambda(0) = \mathbf{x}_\lambda, \end{cases} \tag{1.81}$$

para $\mathbf{x}_\lambda \to \mathbf{x}_0 \in \overline{\operatorname{dom}\phi(0, \cdot)}$, $\partial\phi(0, \mathbf{x}_\lambda) \neq \emptyset$, tienen una única solución $\mathbf{x}_\lambda(\cdot)$ definida en $[0, T[$. Además, cuando $\lambda \to 0^+$, $\mathbf{x}_\lambda(\cdot) \to \mathbf{x}(\cdot)$ y se comprueba que $\mathbf{x}(\cdot)$ es la solución del problema de Cauchy (1.69)+(1.75). Los detalles pueden encontrarse en [10, Chapter II].

Teorema 1.12 (Existencia) *Para cada condición inicial $\mathbf{x}_0 \in \overline{\operatorname{dom}\phi(0, \cdot)}$, se tiene una única solución de (1.69)+(1.75) definida en $[0, T[$. Además:*

(i) Para cada $\varepsilon > 0$, la función $t \rightsquigarrow \phi(t, \mathbf{x}(t))$ es absolutamente continua en cada intervalo compacto contenido en $[\varepsilon, T[$.

(ii) Las funciones $\dot{\mathbf{x}}(t)$, $\sqrt{t}\dot{\mathbf{x}}(t)$ tienen cuadrado integrable en cada intervalo compacto contenido en $[\varepsilon, T[$.

(iii) La función $t \rightsquigarrow t\phi(t, \mathbf{x}(t))$ es esencialmente acotada, es decir, existen $M > 0$ y $A \subset [0, T[$ de medida cero, de forma que $|t\phi(t, \mathbf{x}(t))| \leq M$, para cada $0 \leq t < T$, $t \notin A$.

▶ Ecuación (1.70): caso perturbado. Supongamos que ϕ es un potencial verificando las hipótesis de los apartados anteriores, dependiendo de si es estacionario o varía con el tiempo, y sea $\mathbf{F} : [0, T[\times \mathbb{R}^N \longrightarrow \mathbb{R}^N$ de forma que

(**F**1) Para cada $\mathbf{x} \in \mathbb{R}^N$ fijo, la función $\mathbf{F}(\cdot, \mathbf{x})$ es de cuadrado integrable.

(**F**2) Existe $\gamma : [0, \mathrm{T}[\longrightarrow \mathbb{R}$ de cuadrado integrable, con

$$|\mathbf{F}(t, \mathbf{x}) - \mathbf{F}(t, \mathbf{y})| \leq \frac{\gamma(t)}{2} |\mathbf{x} - \mathbf{y}|$$

Teorema 1.13 (Dependencia continua y unicidad) *Si $\mathbf{x}(\cdot)$, $\mathbf{y}(\cdot)$ son soluciones de (1.70) para las condiciones iniciales $\mathbf{x}_0$, $\mathbf{y}_0$, respectivamente, se verifica la estimación*

$$|\mathbf{x}(t) - \mathbf{y}(t)| \leq |\mathbf{x}_0 - \mathbf{y}_0| \left(e^{\int_0^t \gamma(s) ds} \right)^{1/2}, \quad 0 < t < T. \tag{1.82}$$

De (1.82) es inmediata la dependencia continua respecto de las condiciones iniciales y la unicidad de las soluciones de (1.70) para una condición inicial dada. Es más, puede estimarse la diferencia entre soluciones asociadas a diferentes términos fuente.

Teorema 1.14 (Perturbación) *Sean $\mathbf{x}(\cdot)$, $\mathbf{y}(\cdot)$ soluciones de (1.70) para los términos fuente $\mathbf{F}$ y $\mathbf{G}$, respectivamente, con $\mathbf{x}_0$, $\mathbf{y}_0$ las condiciones iniciales respectivas. Se tiene entonces*

$$|\mathbf{x}(t) - \mathbf{y}(t)|^2 \leq |\mathbf{x}_0 - \mathbf{y}_0|^2 e^{\int_0^t (1+\gamma(s)) ds} + \int_0^t r(s)^2 e^{\int_s^t (1+\gamma(\tau)) d\tau} ds \tag{1.83}$$

donde $r(t) = \sup_{\mathbf{x} \in \mathbb{R}^N} |\mathbf{F}(t, \mathbf{x}) - \mathbf{G}(t, \mathbf{x})|$.

En cuanto a la existencia, las condiciones impuestas sobre el potencial ϕ y el término fuente $\mathbf{F}$ garantizan que el problema

$$\begin{cases} -\mathbf{x}_\lambda(t) = \nabla \phi_\lambda(t, \mathbf{x}_\lambda(t)) + \mathbf{F}(t, \mathbf{x}_\lambda(t)), \\ \mathbf{x}_\lambda(0) = \mathbf{x}_\lambda, \end{cases} \tag{1.84}$$

con $\mathbf{x}_\lambda \to \mathbf{x}_0 \in \overline{\operatorname{dom} \phi(0, \cdot)}$, $\partial \phi(0, \mathbf{x}_\lambda) \neq \emptyset$, tiene una única solución $\mathbf{x}_\lambda(\cdot)$ definida en $[0, T[$ que, como en los casos anteriores, convergerá hacia la solución de nuestro problema cuando $\lambda \to 0^+$ (detalles en [10, Chapter II]).

Teorema 1.15 (Existencia) *Para cada $\mathbf{x}_0 \in \overline{\operatorname{dom} \phi(0, \cdot)}$, existe una única solución de (1.70)+(1.75) definida en $[0, T[$*

► Ecuación (1.72): problema bipotencial. Sean $\phi, \varphi : \mathbb{R}^N \longrightarrow \mathbb{R} \cup \{+\infty\}$ dos potenciales estrictos, convexos e inferiormente semicontinuos, de forma que $\operatorname{dom}\phi \cap \operatorname{dom}\varphi \neq \emptyset$. Se asumen además las siguientes hipótesis adicionales relacionadas con el segundo potencial:

(φ1) $\partial\phi(\mathbf{x}) \subseteq \partial\varphi(\mathbf{x})$, para todo $\mathbf{x}$.

(φ2) Existen $0 < k < 1$, $\eta : \mathbb{R} \longrightarrow \mathbb{R}_+$ creciente de forma que, para cada $\mathbf{x} \in \operatorname{dom}\phi$:

$$|\partial\varphi^0(\mathbf{x})| \leq k|\partial\phi^0(\mathbf{x})| + \eta(\phi(\mathbf{x}) + |\mathbf{x}|) \tag{1.85}$$

donde $\partial\phi^0(\mathbf{x}) = \operatorname{proj}_{\partial\phi(\mathbf{x})}(\mathbf{0})$.

(φ3) Se tiene $c > 0$ tal que

$$\varphi(\mathbf{x}) \leq k\phi(\mathbf{x}) + c \tag{1.86}$$

para cada $\mathbf{x} \in \operatorname{dom}\phi$.

(φ4) Para cada $r > 0$ existen una constante $K_r > 0$ y una función creciente $\rho_r : \mathbb{R}_+ \longrightarrow \mathbb{R}_+$ de forma que si $\mathbf{x}, \mathbf{y} \in B_r(\mathbf{0})$ son puntos de subsiferenciabilidad de ϕ, para cada $\mathbf{u} \in \partial\varphi(\mathbf{x})$, $\mathbf{v} \in \partial\varphi(\mathbf{y})$ se verifica la desigualdad

$$\langle \mathbf{v} - \mathbf{u}, \mathbf{y} - \mathbf{x} \rangle \leq \rho_r(|\mathbf{x}| + |\mathbf{y}|)\,(\phi(\mathbf{x}) + \phi(\mathbf{y}) + K_r)\,|\mathbf{x} - \mathbf{y}|^2 \tag{1.87}$$

Finalmente, la función $\beta : [0, +\infty[\longrightarrow \mathbb{R}$ es continua.

Usando (φ4) puede estimarse la distancia entre dos soluciones de (1.72), lo que permite obtener la dependencia continua de la condición inicial y la unicidad del problema de Cauchy asociado.

Teorema 1.16 (Dependencia continua y unicidad) *Si $\mathbf{x}(\cdot)$, $\mathbf{y}(\cdot)$ son soluciones de (1.72) para las condiciones iniciales $\mathbf{x}_0, \mathbf{y}_0$, respectivamente, para cada $0 < \tau < T$, se tiene una constante $M(\tau) > 0$ de forma que*

$$|\mathbf{x}(t) - \mathbf{y}(t)| \leq M(\tau)|\mathbf{x}_0 - \mathbf{y}_0|e^{\frac{1}{2}\int_0^t |\beta(s)|ds}, \quad 0 \leq t \leq \tau. \tag{1.88}$$

En cuanto a la existencia de solución, dado $\mathbf{x}_0 \in \overline{dom\phi \cap \operatorname{dom}\varphi}$, si $\mathbf{x}_\lambda \to \mathbf{x}_0$ cuando $\lambda \to 0^+$, con $\partial\phi(\mathbf{x}_\lambda) \neq \emptyset$, $\partial\varphi(\mathbf{x}_\lambda) \neq \emptyset$, los problemas de Cauchy

$$\begin{cases} -\dot{\mathbf{x}}_\lambda(t) = \nabla\phi_\lambda(\mathbf{x}_\lambda(t)) + \beta(t)\nabla\varphi_\lambda(\mathbf{x}_\lambda(t)), \\ \mathbf{x}_\lambda(0) = \mathbf{x}_\lambda, \end{cases} \tag{1.89}$$

tienen una única solución $\mathbf{x}_\lambda(\cdot)$ definida en $[0, +\infty[$. Además, las hipótesis sobre el segundo potencial, (φ1)-(φ4), garantizan la convergencia de $\mathbf{x}_\lambda(\cdot)$ hacia una función $\mathbf{x}(\cdot)$ que es solución de (1.72)+(1.75) (los detalles pueden encontrarse en [1] y [10]).

Teorema 1.17 (Existencia) *Para cada $\mathbf{x}_0 \in \overline{\operatorname{dom}\phi \cap \operatorname{dom}\varphi}$, existe una única solución de (1.72)+(1.75) en el intervalo $[0, +\infty[$.*

Finalmente, si $\beta \geq 0$ y se verifican ciertas condiciones adicionales de carácter técnico, se tienen resultados sobre el comportamiento asintótico de las soluciones de (1.72).

Teorema 1.18 (Theorem 3.1 en [1]) *Supongamos que* $\operatorname{argmin}\varphi \neq \emptyset$ *y que existe el mínimo de* ϕ *en este conjunto, es decir,*

$$\operatorname{argmin}_{\operatorname{argmin}\varphi}\phi = \left\{\mathbf{x} \in \operatorname{argmin}\varphi : \phi(\mathbf{x}) = \inf_{\mathbf{z}\in\operatorname{argmin}\varphi}\phi(\mathbf{z})\right\} \neq \emptyset$$

y se verifican las condiciones

(H1) Si $\mathbf{u} \in \partial\psi_{\operatorname{argmin}\varphi}(\mathbf{x})$*, para algún* $\mathbf{x} \in \operatorname{argmin}\varphi$*,*

$$\int_0^{+\infty} \beta(t)\left[\varphi^*(\mathbf{u}/\beta(t)) - \sigma_{\operatorname{argmin}\varphi}(\mathbf{u}/\beta(t))\right] dt < +\infty$$

donde $\psi_{\operatorname{argmin}\varphi}$ *y* $\sigma_{\operatorname{argmin}\varphi}$ *son las funciones indicatriz y soporte, respectivamente, del conjunto* $\operatorname{argmin}\varphi$ *y* φ^* *es la conjugada de Fenchel-Moreau*[4] *del potencial* φ*.*

(H2) $\beta : \mathbb{R}_+ \longrightarrow \mathbb{R}_+$ *es de clase* C^1*,* $\beta(t) \to +\infty$ *cuando* $t \to +\infty$ *y se tienen* $a \geq 0$*,* $t_0 \geq 0$ *de forma que*

$$0 \leq \dot{\beta}(t) \leq a\beta(t), \quad t \geq t_0.$$

Se tiene entonces que, si $\mathbf{x}(\cdot)$ *es solución de (1.72):*

(i) (Convergencia asintótica) Existe el límite $\lim_{t\to+\infty}\mathbf{x}(t) \in \operatorname{argmin}_{\operatorname{argmin}\varphi}\phi$*.*

(ii) (Trayectoria minimizante) Existen los límites

$$\lim_{t\to+\infty}\varphi(\mathbf{x}(t)) = 0,$$

$$\lim_{t\to+\infty}\phi(\mathbf{x}(t)) = \min_{\mathbf{z}\in\operatorname{argmin}\varphi}\phi(\mathbf{z})$$

$$\lim_{t\to+\infty}|\mathbf{x}(t) - \mathbf{z}|, \text{ para cada } \mathbf{z} \in \operatorname{argmin}\varphi$$

(iii) $\lim_{t\to+\infty}\beta(t)\varphi(\mathbf{x}(t)) = 0$*,* $\lim_{t\to+\infty}\int_0^t \beta(s)\varphi(\mathbf{x}(s))\,ds < +\infty$

[4] La conjugada de Fenchel-Moreau de una función convexa φ se define como

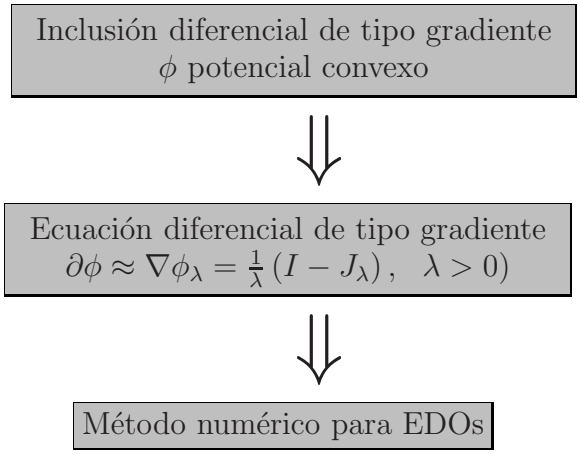
$$\varphi^*(\mathbf{v}) = \sup_{\mathbf{x}\in\mathbb{R}^N}\left(\langle\mathbf{v},\mathbf{x}\rangle - \varphi(\mathbf{x})\right).$$

Capítulo 2

Algoritmos y códigos

El método de aproximación considerado, que denominamos de *Moreau-Yosida*, se basa en aproximar el término donde aparece la subdiferencial por su regularización de Yosida, que es un término univaluado que en este caso coincide con el gradiente de la envoltura de Moreau del potencial. Con ello se obtiene una ecuación diferencial ordinaria cuya solución a su vez podemos calcular de forma aproximada por cualquiera de los múltiples métodos disponibles.

Aquí vemos un diagrama del método:



Los parámetros $\lambda, h > 0$ indican, respectivamente, el nivel de aproximación de la regularización de Yosida de la subdiferencial y el tamaño de la discretización asociada al método numérico para ecuaciones diferenciales ordinarias. Así, para valores de λ, h pequeños, las solución obtenida en $(\mathcal{P}_{\lambda,h})$ proporciona una aproximación a la solución del problema original $(\mathcal{P})$.

En este capítulo se presentan diferentes algoritmos programados con el software MATLAB que utilizan el método de Moreau-Yosida para aproximar la solución de distintos tipos de problema. Los métodos numéricos utilizados para

aproximar la solución de los problemas $(\mathcal{P}_\lambda)$ han sido los de Euler y Runge-Kutta de orden cuatro, abreviadamente RK4.

Se ha tratado de incluir, para mayor comodidad y sencillez, todos los posibles casos de resolución que veremos a continuación en un sólo programa para cada método numérico. Además, en el aspecto de programación, hay que destacar que al trabajar con vectores sin longitud predefinida, estos programas son válidos para problemas de cualquier dimensión con la que se quiera trabajar, lo que los hace altamente flexibles.

Se presentará por separado cada caso estudiado indicando su formulación matemática y el algoritmo y código utilizado para su resolución. Los códigos completos están incluidos en el Anexo de esta memoria.

Se presentan así mismo ejemplos concretos resueltos mediante la implementación de los códigos MATLAB para ilustrar su funcionamiento.

2.1 Caso básico

Se considera en primer lugar el problema de determinar el flujo asociado al gradiente de un potencial convexo no necesariamente diferenciable a partir de una condición inicial. El potencial puede variar con el tiempo. Tenemos, pues, el problema de la forma:

$$\left.\begin{array}{l} -\dot{\mathbf{x}}(t) \in \partial\phi(t, \mathbf{x}(t)) \\ \mathbf{x}(0) = \mathbf{x}^0 \end{array}\right\} \qquad (\mathcal{P})$$

con $\phi : [0, T[\times \mathbb{R}^N \longrightarrow \mathbb{R}$ convexa en la segunda variable, $0 < T \leq +\infty$. Tomando $\lambda > 0$ y aplicando la regularización de Yosida, queda el problema aproximado:

$$\left.\begin{array}{l} -\dot{\mathbf{x}}_\lambda(t) = \frac{1}{\lambda}\left(\mathbf{x}_\lambda(t) - J_\lambda(t, \mathbf{x}_\lambda(t))\right) \\ \mathbf{x}_\lambda(0) = \mathbf{x}^0 \end{array}\right\} \qquad (\mathcal{P}_\lambda)$$

con

$$J_\lambda(t, \mathbf{x}) = \underset{\mathbf{y}\in\mathbb{R}^N}{\operatorname{argmin}} \left(\phi(t, \mathbf{y}) + \frac{1}{2\lambda}\left|\mathbf{x} - \mathbf{y}\right|^2\right)$$

el campo resolvente. Aplicamos a este problema los métodos de aproximación de Euler y RK4 para obtener el problema discretizado $(\mathcal{P}_{\lambda,h})$.

2.1.1 Método de Euler

Usando el método de Euler para ecuaciones diferenciales se obtiene el siguiente algoritmo de aproximación para las soluciones de la ecuación regularizada $(\mathcal{P}_\lambda)$ en el intervalo $[0, T]$.

Algoritmo de Euler

- $\lambda > 0,\ m \geq 1$

  1. Inicialización: $\mathbf{x}_\lambda^{m,0} = \mathbf{x}^0$

  $$h_m = T/m$$

  $$j = 0$$

  2. Cálculo de la resolvente:

  $$\mathbf{y}_\lambda^{m,j} = \underset{\mathbf{y}\in\mathbb{R}^N}{\operatorname{argmin}} \left(\phi(t^{m,j}, \mathbf{y}) + \frac{1}{2\lambda} \left| \mathbf{y} - \mathbf{x}_\lambda^{m,j} \right|^2 \right)$$

  3. Definición del nuevo nodo:

  $$\mathbf{x}_\lambda^{m,j+1} = \mathbf{x}_\lambda^{m,j} - \frac{h_m}{\lambda} \left(\mathbf{x}_\lambda^{m,j} - \mathbf{y}_\lambda^{m,j} \right)$$

  4. Actualización: $j = j + 1$

  5. Si $j < m$ ir a **2.**, en otro caso **Fin**.

El código para la resolución del problema que hemos llamado básico mediante el método de Euler se ha programado en la función **euler.m** incluida en el Anexo. En realidad, como se ha indicado antes, se han programado todos los tipos de problemas dentro de la misma función por comodidad por lo que para seleccionar el tipo de problema a resolver se hará introduciendo los argumentos de entrada adecuados.

Se van a explicar en primer lugar, por tanto, los argumentos de entrada y salida de la función. La función **euler.m** tiene la siguiente forma:

```
function [x,d]=euler(fun_prueba,x0,a,b,m,lambda,...
    term_fuente,fun_prueba_2,mu,beta,dominiok)
```

Sus argumentos de entrada son:

- `fun_prueba`: puntero que indica la dirección de la función potencial a utilizar en el problema. Hay que destacar que este potencial puede ser tanto constante como variable con el tiempo, ya que la función está preparada para trabajar con cualquier caso.

- `x0`: vector que contiene los valores iniciales de la variable utilizada dados por las condiciones iniciales del problema. Hay que señalar que, como se ha indicado anteriormente, esta función está programada para trabajar con vectores y matrices por lo que no existe ninguna limitación en cuanto a las dimensiones de las variables implicadas en el problema. Sin embargo, por tema de visualización se ha preparado para trabajar hasta con tres dimensiones espaciales, ya que son las máximas que se pueden representar.

- `a`: extremo inferior del intervalo temporal en el que se quiere obtener la solución.
- `b`: extremo superior del intervalo temporal en el que se quiere obtener la solución.
- `m`: número de pasos en los que dividir el intervalo temporal.
- `lambda`: nivel de aproximación de la envoltura de Moreau.

El resto de argumentos de entrada se definirán ahora pero no se usan para este tipo de problema, sino para los que se van a tratar más adelante:

- `term_fuente`: puntero que indica la dirección de la función perturbación utilizada en los problemas con término fuente.
- `fun_prueba_2`: puntero que indica la dirección de la segunda función potencial utilizada en los problemas bipotenciales.
- `mu`: nivel de aproximación de la envoltura de Moreau del segundo potencial en el caso de problemas bipotenciales.
- `beta`: valor de β que multiplica a la segunda función potencial en los problemas bipotenciales.
- `dominiok`: si en este campo hay un puntero indica que se trata de un problema de persecución, por lo que se obvia la función `fun_prueba` y se considera como función potencial la distancia al conjunto definido.

Los argumentos de salida son:

- `x`: vector o matriz que contiene la solución en cada instante de tiempo. Esta matriz tendrá tantas filas como subdivisiones temporales se hayan utilizado y tantas columnas como dimensiones tenga el problema considerado. Las dimensiones del problema vendrán determinadas por la variable `dim` que se definirá como una variable de tipo `global` ya que será necesaria en diferentes funciones usadas.
- `d`: vector que contiene la distancia de la solución al conjunto definido en cada instante del tiempo. Este argumento de salida sólo tendrá sentido cuando se trate de un problema de persecución como se verá más adelante.

Por tanto, para el caso básico solo habría que indicar la función a utilizar en `fun_prueba`, y el valor de `x0`, `a`, `b`, `m` y `lambda`, dejando en blanco el resto de argumentos de entrada.

El funcionamiento de la función sigue básicamente el esquema presentado en el pseudocódigo al comienzo de la sección. En primer lugar con los extremos proporcionados del intervalo temporal y el número de pasos deseado se obtiene el tamaño de paso h y con esto se crea un vector de tiempos `T`. Seguidamente se crea una matriz de ceros de `m+1` filas y `dim` columnas donde se irán almacenando

los valores de la solución para cada instante de tiempo. Se rellenan entonces los valores de la primera fila de la matriz con los valores dados por las condiciones iniciales del problema (`x0`).

Una vez que se ha hecho la inicialización se entra en un bucle *for* que se va recorriendo desde el principio al final del intervalo temporal indicado, avanzando con tamaño de paso h (en cada iteración el instante temporal considerado está definido por la variable `t`). En él está programado el algoritmo propiamente dicho. En un solo paso se realiza la definición del nuevo nodo, calculando la resolvente mediante la función **moreau.m**.

Esta función tiene la siguiente forma:

```
function f=moreau(t,x,x0,lambda,fun_prueba)
```

siendo sus argumentos de entrada:

- `t`: escalar con el valor del instante de tiempo considerado.
- `x`: vector que contiene el valor de la solución en el instante `t`.
- `x0`: vector de valores iniciales de la solución.
- `lambda`: nivel de aproximación de la envoltura de Moreau.
- `fun_prueba`: puntero que indica la dirección de la función potencial del problema.

Como argumento de salida devuelve un vector con el valor de las diferentes componentes de la resolvente.

La función **moreau.m**, como se ha comentado anteriormente, se utiliza para calcular la resolvente, en la cual hay que obtener el mínimo de una expresión. Este mínimo se obtiene con la función que trae integrada MATLAB llamada `fminsearch`, cuyos argumentos de entrada deben ser la función a la cual queremos buscar su mínimo (ver paso **2.** del algoritmo) y el punto donde se comenzará a buscarlo (en este caso `x0`). Dicha función devuelve el punto donde se alcanza el mínimo y el valor de la función en dicho punto. En este caso, sólo interesa el punto donde se alcanza el mínimo, por lo que únicamente se almacenará dicho vector. Es importante indicar que la función `fminsearch` utiliza el algoritmo de Lagarias, que no usa gradientes numéricos ni analíticos para buscar el mínimo, por lo que funciona con funciones no diferenciables (como es nuestro caso). Además, al ser la expresión a mimimizar convexa no tiene mínimos locales (ver Proposición 1.10 en el capítulo anterior) y el mínimo global existe y es único (Teorema 1.6 de existencia y unicidad de la resolvente), lo que garantiza el óptimo funcionamiento de la función `fminsearch`.

Una vez que se termina el bucle *for* de la función **euler.m**, el algoritmo se para puesto que se ha llegado a la solución aproximada. La variable `x` contiene los valores de dicha solución para cada instante de la discretización temporal.

Comentar por último que se ha añadido dentro del algoritmo una barra de progreso que va avanzando en cada iteración del buble para saber en cada momento cuánto tiempo queda para obtener la solución aproximada.

2.1.2 Método de RK4

Si se utiliza el método RK4 para aproximar numéricamente la solución de $(\mathcal{P}_\lambda)$ se tiene el siguiente pseudocódigo:

Algoritmo RK4

- $\lambda > 0,\ m \geq 1$

  1. Inicialización: $\mathbf{x}_\lambda^{m,0} = \mathbf{x}^0$

  $$h_m = T/m$$

  $$j = 0$$

  2. Cálculo de los coeficientes:

  $$\mathbf{k}_{\lambda,1}^{m,j} = \frac{1}{\lambda}\left(\mathbf{x}_\lambda^{m,j} - J_\lambda(t^{m,j}, \mathbf{x}_\lambda^{m,j})\right)$$

  $$\mathbf{k}_{\lambda,2}^{m,j} = \frac{1}{\lambda}\left(\mathbf{x}_\lambda^{m,j} - \frac{h_m}{2}\mathbf{k}_{\lambda,1}^{m,j} - J_\lambda(t^{m,j} + \frac{h_m}{2}, \mathbf{x}_\lambda^{m,j} - \frac{h_m}{2}\mathbf{k}_{\lambda,1}^{m,j})\right)$$

  $$\mathbf{k}_{\lambda,3}^{m,j} = \frac{1}{\lambda}\left(\mathbf{x}_\lambda^{m,j} - \frac{h_m}{2}\mathbf{k}_{\lambda,2}^{m,j} - J_\lambda(t^{m,j} + \frac{h_m}{2}, \mathbf{x}_\lambda^{m,j} - \frac{h_m}{2}\mathbf{k}_{\lambda,2}^{m,j})\right)$$

  $$\mathbf{k}_{\lambda,4}^{m,j} = \frac{1}{\lambda}\left(\mathbf{x}_\lambda^{m,j} - h_m\mathbf{k}_{\lambda,3}^{m,j} - J_\lambda(t^{m,j} + h_m, \mathbf{x}_\lambda^{m,j} - h_m\mathbf{k}_{\lambda,3}^{m,j})\right)$$

  donde

  $$J_\lambda(t, \mathbf{z}) = \underset{\mathbf{y}\in\mathbb{R}^N}{\operatorname{argmin}}\left(\phi(t, \mathbf{y}) + \frac{1}{2\lambda}\left|\mathbf{y} - \mathbf{z}\right|^2\right)$$

  3. Definición del nuevo nodo:

  $$\mathbf{x}_\lambda^{m,j+1} = \mathbf{x}_\lambda^{m,j} - \frac{h_m}{6}\left(\mathbf{k}_{\lambda,1}^{m,j} + 2\mathbf{k}_{\lambda,2}^{m,j} + 2\mathbf{k}_{\lambda,3}^{m,j} + \mathbf{k}_{\lambda,4}^{m,j}\right)$$

  4. Actualización: $j = j + 1$

  5. Si $j < m$ ir a **2**, en otro caso **Fin**.

Este algoritmo se ha programado en la función **rk4.m**:

```
function [x,d]=rk4(fun_prueba,x0,a,b,m,lambda,...
   term_fuente,fun_prueba_2,mu,beta,dominiok)
```

Como se puede observar los argumentos de entrada y salida son los mismos que en la función **euler.m**, por lo que no hace falta volverlos a explicar.

Su funcionamiento también es análogo, la única diferencia radica en la expresión de los coeficientes, por lo que en **rk4.m** se utilizan las fórmulas indicadas en el pseudocódigo, para lo cual en cada iteración se evalúa cuatro veces la función **moreau.m**. Por lo demás es válido todo lo indicado para el método de Euler.

2.1.3 Ejemplos de aplicación: caso básico

Veamos ahora cómo funciona nuestro código con algunos ejemplos concretos.

Ejemplo 2.1 El algoritmo de Moreau-Yosida puede usarse no sólo para inclusiones subdiferenciales, sino para resolver ecuaciones diferenciales ordinarias de tipo gradiente

$$\dot{\mathbf{x}}(t) = \nabla\phi(t, \mathbf{x}(t)) \tag{2.3}$$

donde el potencial convexo ϕ es diferenciable, pero el cálculo de su gradiente es tedioso o complicado. En este caso el algoritmo de Moreau-Yoside permite obviar dicho cálculo.

Por ejemplo, dado el conjunto convexo

$$C = \left\{\mathbf{x} \in \mathbb{R}^N : |\mathbf{x} - \mathbf{c}_0| \leq \delta\right\}$$

podemos definir el cuadradro de la función distancia, que en este caso es de la forma,

$$d_C^2(\mathbf{x}) = \begin{cases} 0 & \text{si } |\mathbf{x} - \mathbf{c}_0| \leq \delta \\ (\delta - |\mathbf{x} - \mathbf{c}_0|)^2 & \text{si } |\mathbf{x} - \mathbf{z}_0| > \delta \end{cases}$$

y proponer el siguiente potencial convexo y diferenciable

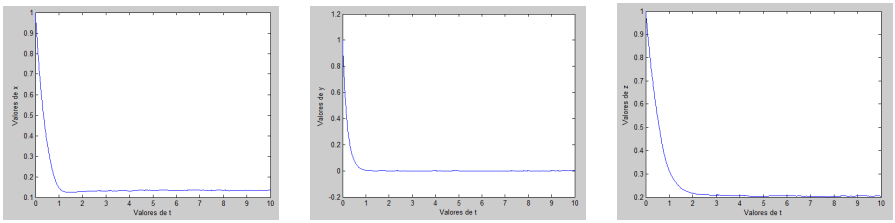
$$\phi(\mathbf{x}) = e^{d_C^2(\mathbf{x})} + \mathbf{x}^t Q \mathbf{x}$$

siendo Q una matriz cuadrada, simétrica y definida positiva. Se ha usado el código MATLAB del algoritmo de Moreau-Yosida para resolver el sistema (2.3) para el potencial anterior en el caso $N = 3$, $\delta = 1$, $\mathbf{c}_0 = (1, 0, 1)$ y la matriz

$$Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2.5 & 0 \\ 0 & 0 & 0.6 \end{pmatrix}$$

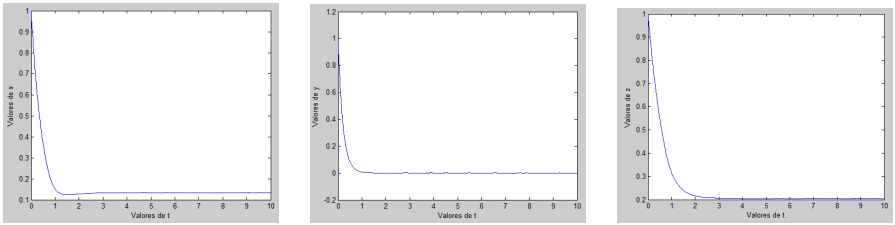
Se muestran las gráficas de los resultados obtenidos mediante los métodos de Euler y RK4. En las simulaciones se han usado los siguientes parámetros:

Parámetro	Valor	Parámetro	Valor	Parámetro	Valor
a	0	m	1000	x0	1
b	10	lambda	0.001	y0	1
				z0	1



$x(t)$ $y(t)$ $z(t)$

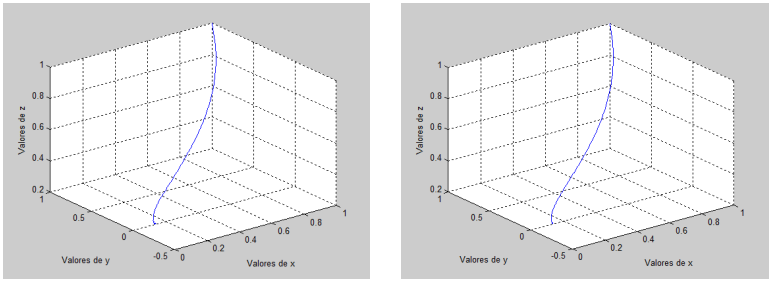
Algoritmo Euler



$x(t)$ $y(t)$ $z(t)$

Algoritmo RK4

Figura 2.1: Componentes de la solución

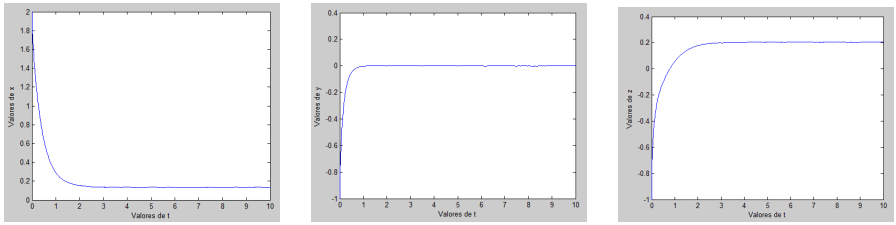


Euler Runge-Kutta 4

Figura 2.2: Trayectoria $(x(t), y(t), z(t))$

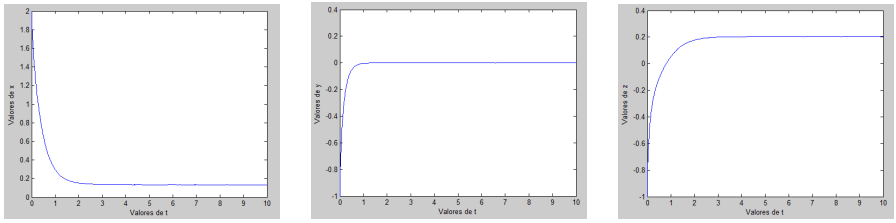
Repetimos la simulación usando otro punto inicial obteniendo los siguientes resultados:

Parámetro	Valor	Parámetro	Valor	Parámetro	Valor
a	0	m	1000	x0	2
b	10	lambda	0.001	y0	-1
				z0	-1



$x(t)$ $y(t)$ $z(t)$

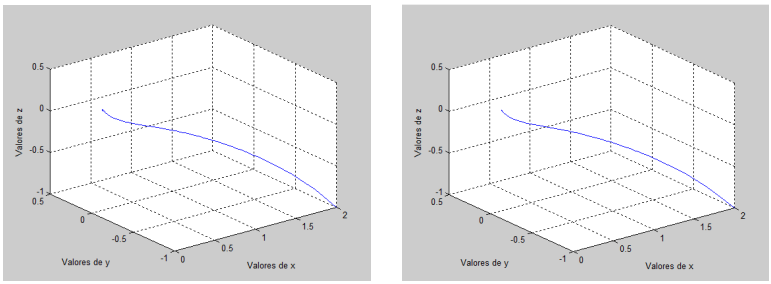
Algoritmo Euler



$x(t)$ $y(t)$ $z(t)$

Algoritmo RK4

Figura 2.3: Componentes de la solución



Euler Runge-Kutta 4

Figura 2.4: Trayectoria $(x(t), y(t), z(t))$

Ejemplo 2.2 Para poner ahora un ejemplo de un potencial que dependa del tiempo, tomamos dos funciones convexas en las variables x, y:

$$\phi_1(t,x,y) = (\cos(t)x)^2 + e^y$$
$$\phi_2(t,x,y) = (x,y)\begin{pmatrix} 2+\operatorname{sen}(t) & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

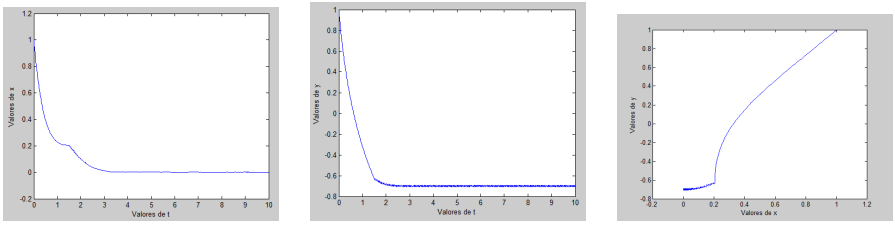
y definimos el potencial del problema como el máximo de ambas

$$\phi(t,x,y) = \max\left(\phi_1(t,x,y), \phi_2(t,x,y)\right)$$

por lo que no será diferenciable. Resolveremos, pues, la inclusión difernciał $(\mathcal{P})$ de la página 46. Al igual que antes, se mostrará la solución obtenida con el método de Euler y con el RK4 para dos valores iniciales distintos.

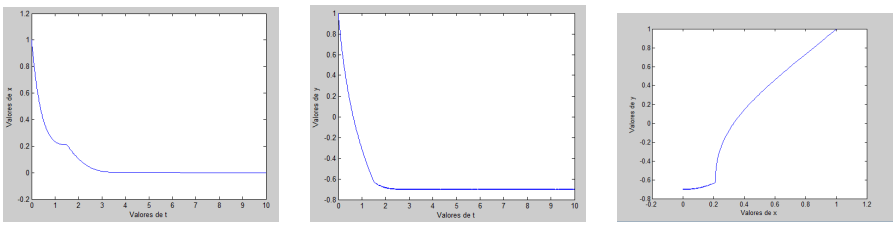
En primer lugar usamos los siguientes parámetros:

Parámetro	Valor	Parámetro	Valor
a	0	lambda	0.001
b	10	x0	1
m	1000	y0	1



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.5: Algoritmo Euler

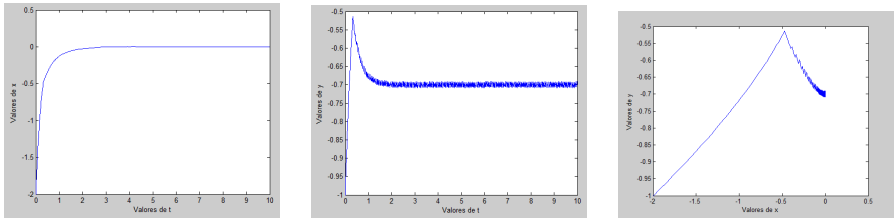


$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.6: Algoritmo RK4

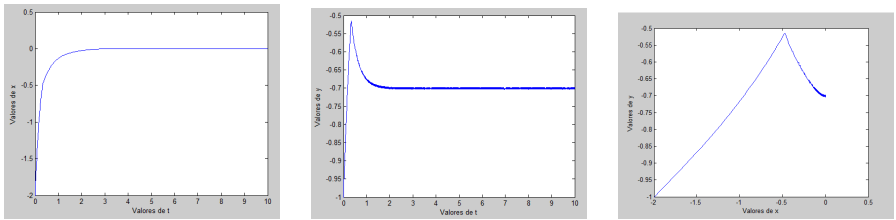
Seguidamente se cambia el punto inicial, manteniendo el resto de parámetros:

Parámetro	Valor	Parámetro	Valor
a	0	lambda	0.001
b	10	x0	-2
M	1000	y0	-1



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.7: Algoritmo Euler



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.8: Algoritmo RK4

2.2 Caso con término fuente

La inclusión estudiada en el apartado anterior puede perturbarse añadiendo un término fuente dado por un campo vectorial $\mathbf{F} : [0, +\infty[\times \mathbb{R}^N \longrightarrow \mathbb{R}^N$ continuo y de clase C^1 en la segunda variable. El problema $(\mathcal{P})$ queda entonces (en la forma más general en que el potencial varía con el tiempo) de la forma

$$\left.\begin{array}{l} -\dot{\mathbf{x}}(t) \in \partial\phi(t, \mathbf{x}(t)) - \mathbf{F}(t, \mathbf{x}(t)) \\ \mathbf{x}(0) = \mathbf{x}^0 \end{array}\right\} \qquad (\mathcal{P})$$

Tomando $\lambda > 0$ y aplicando la regularización de Yosida, queda:

$$\left.\begin{array}{l} -\dot{\mathbf{x}}_\lambda(t) = \frac{1}{\lambda}\left(\mathbf{x}_\lambda(t) - J_\lambda(t, \mathbf{x}_\lambda(t))\right) - \mathbf{F}(t, \mathbf{x}_\lambda(t)) \\ \mathbf{x}_\lambda(0) = \mathbf{x}^0 \end{array}\right\} \qquad (\mathcal{P}_\lambda)$$

Aplicamos a este problema los métodos de aproximación de Euler y RK4 para obtener el problema discretizado $(\mathcal{P}_{\lambda,h})$.

2.2.1 Método de Euler

En el algoritmo no habrá que modificar la parte del cálculo de la resolvente, que no se ve afectado por el término fuente. En realidad solamente es necesario cambiar el tercer paso en que se obtiene el nuevo nodo teniendo en cuenta el campo $\mathbf{F}$.

Modificación Euler (potencial variable+término fuente)

---

3. Definición del nuevo nodo:

$$\mathbf{x}_\lambda^{m,j+1} = \mathbf{x}_\lambda^{m,j} - \frac{h_m}{\lambda}\left(\mathbf{x}_\lambda^{m,j} - \mathbf{y}_\lambda^{m,j}\right) + h_m \mathbf{F}(t^{m,j}, \mathbf{x}_\lambda^{m,j})$$

Se utiliza para la resolución de este problema de nuevo la función **euler.m** a la que ahora se le añade el argumento de entrada `term_fuente` en el que se define, como se ha explicado anteriormente, el puntero del campo vectorial que genera la perturbación. La función tiene el mismo funcionamiento que en el caso anterior en cuanto a la inicialización de las variables, solo que ahora dentro del bucle *for* se añade el término fuente evaluado en el instante de tiempo dado y en el valor de la solución en el instante considerado.

Al igual que antes, una vez que se ha terminado de recorrer el bucle *for*, la función devuelve el vector o la matriz con los valores de la solución para cada instante de tiempo.

2.2.2 Método de RK4

En el algoritmo asociado al método de Runge-Kutta de orden 4 hay que modificar sustancialmente el cálculo de los coeficientes del segundo paso, ya que dependen no sólo del potencial, si no también del campo $\mathbf{F}$. Se introducen para ello unos coeficientes auxiliares que ligan la envoltura de Moreau con el término fuente.

Modificación RK4 (potencial variable+término fuente)

2. Cálculo de los coeficientes:

$$\mathbf{k}_{\lambda,1}^{m,j} = \frac{1}{\lambda}\left(\mathbf{x}_\lambda^{m,j} - J_\lambda(t^{m,j}, \mathbf{x}_\lambda^{m,j})\right)$$

$$\mathbf{q}_{\lambda,1}^{m,j} = \mathbf{F}(t^{m,j}, \mathbf{x}_\lambda^{m,j}) - \mathbf{k}_{\lambda,1}^{m,j}$$

$$\mathbf{k}_{\lambda,2}^{m,j} = \frac{1}{\lambda}\left(\mathbf{x}_\lambda^{m,j} + \frac{h_m}{2}\mathbf{q}_{\lambda,1}^{m,j} - J_\lambda(t^{m,j} + \frac{h_m}{2}, \mathbf{x}_\lambda^{m,j} + \frac{h_m}{2}\mathbf{q}_{\lambda,1}^{m,j})\right)$$

$$\mathbf{q}_{\lambda,2}^{m,j} = \mathbf{F}\left(t^{m,j} + \frac{h_m}{2}, \mathbf{x}_\lambda^{m,j} + \frac{h_m}{2}\mathbf{q}_{\lambda,1}^{m,j}\right) - \mathbf{k}_{\lambda,2}^{m,j}$$

$$\mathbf{k}_{\lambda,3}^{m,j} = \frac{1}{\lambda}\left(\mathbf{x}_\lambda^{m,j} + \frac{h_m}{2}\mathbf{q}_{\lambda,2}^{m,j} - J_\lambda(t^{m,j} + \frac{h_m}{2}, \mathbf{x}_\lambda^{m,j} + \frac{h_m}{2}\mathbf{q}_{\lambda,2}^{m,j})\right)$$

$$\mathbf{q}_{\lambda,3}^{m,j} = \mathbf{F}\left(t^{m,j} + \frac{h_m}{2}, \mathbf{x}_\lambda^{m,j} + \frac{h_m}{2}\mathbf{q}_{\lambda,2}^{m,j}\right) - \mathbf{k}_{\lambda,3}^{m,j}$$

$$\mathbf{k}_{\lambda,4}^{m,j} = \frac{1}{\lambda}\left(\mathbf{x}_\lambda^{m,j} + h_m\mathbf{q}_{\lambda,3}^{m,j} - J_\lambda(t^{m,j} + h_m, \mathbf{x}_\lambda^{m,j} + h_m\mathbf{q}_{\lambda,3}^{m,j})\right)$$

$$\mathbf{q}_{\lambda,4}^{m,j} = \mathbf{F}\left(t^{m,j} + h_m, \mathbf{x}_\lambda^{m,j} + h_m\mathbf{q}_{\lambda,3}^{m,j}\right) - \mathbf{k}_{\lambda,4}^{m,j}$$

donde

$$J_\lambda(t, \mathbf{z}) = \underset{\mathbf{y}\in\mathbb{R}^N}{\operatorname{argmin}}\left(\phi(t, \mathbf{y}) + \frac{1}{2\lambda}|\mathbf{y} - \mathbf{z}|^2\right)$$

3. Definición del nuevo nodo:

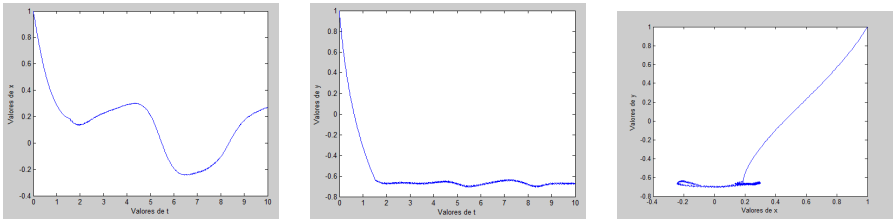
$$\mathbf{x}_\lambda^{m,j+1} = \mathbf{x}_\lambda^{m,j} + \frac{h_m}{6}\left(\mathbf{q}_{\lambda,1}^{m,j} + 2\mathbf{q}_{\lambda,2}^{m,j} + 2\mathbf{q}_{\lambda,3}^{m,j} + \mathbf{q}_{\lambda,4}^{m,j}\right)$$

En el aspecto de programación, análogamente a como se ha hecho en el método de Euler, bastará con añadir el argumento de entrada `term_fuente` en la función **rk4.m** para que sea capaz de resolver correctamente este tipo de problema. Al igual que antes la única parte que varía es dentro del bucle *for*, ya que en el cálculo de los coeficientes hay que añadir ahora a cada uno de ellos el término fuente evaluado en el instante de tiempo y el valor de la solución adecuados.

2.2.3 Ejemplos de aplicación: caso con término fuente

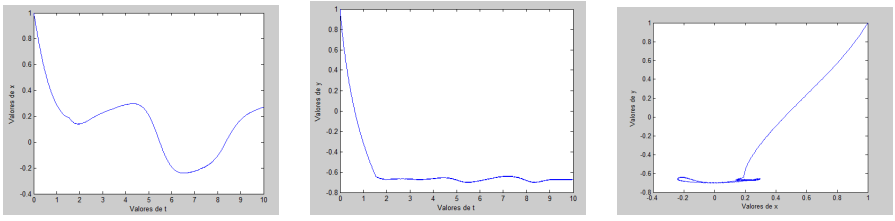
Ejemplo 2.3 Se considera ahora el problema del Ejemplo 2.2 y se le añade una perturbación dada por el campo $\mathbf{F}(t, x, y) = (\cos(ty), 0)$. Manteniendo igual el resto de parámetros, observamos las diferencias que se obtienen en la solución. Cabe destacar que, aunque el término fuente solamente afecta a la primera componente, al estar ambas acopladas se observan cambios en las dos componentes de la solución.

Parámetro	Valor	Parámetro	Valor
a	0	lambda	0.001
b	10	x0	1
m	1000	y0	1



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.9: Algoritmo Euler

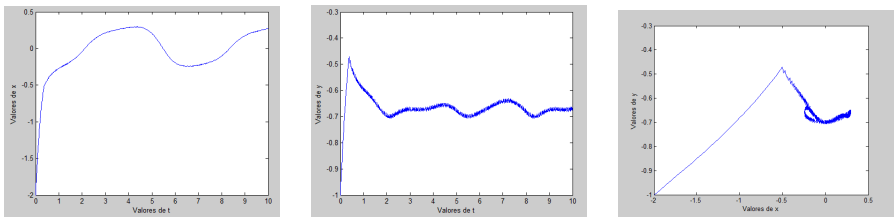


$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.10: Algoritmo RK4

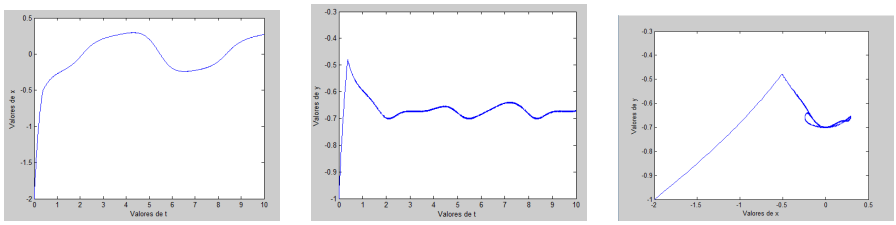
Calculamos ahora la solución partiendo de un punto distinto:

Parámetro	Valor	Parámetro	Valor
a	0	lambda	0.001
b	10	x0	-2
m	1000	y0	-1



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.11: Algoritmo Euler



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.12: Algoritmo RK4

En este caso, al igual que en el Ejemplo 2.2, se observa cómo el método RK4 amortigua las oscilaciones en la segunda componente, mejorando la precisión.

Ejemplo 2.4 Para mostrar la influencia del término fuente sobre la solución se va a mostrar la diferencia que aparece por su inclusión o no en un problema donde el potencial es una función distancia (este tipo de problemas se han desarrollado en más profundidad en el Capítulo 4).

Dado el conjunto $C = \left\{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\right\}$ se considera la inclusión subdiferencial

$$-\dot{\mathbf{x}}(t) \in \partial d_C(\mathbf{x}(t))$$

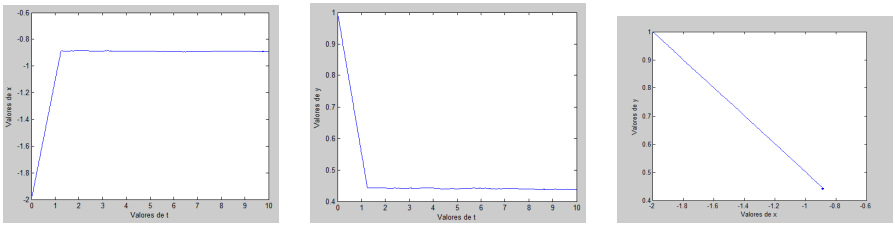
donde la función distancia d_C tiene en este caso particular la forma

$$d_C(x, y) = \begin{cases} 0 & \text{si } x^2 + y^2 \leq 1 \\ \sqrt{x^2 + y^2} - 1 & \text{si } x^2 + y^2 > 1 \end{cases} \tag{2.6}$$

Resolveremos el problema (2.6) usando el método de Euler para distintas condiciones iniciales usando los siguientes parámetros:

Parámetro	Valor	Parámetro	Valor
a	0	lambda	0.001
b	10	x0	-2
m	1000	y0	1

Se obtiene la siguiente solución

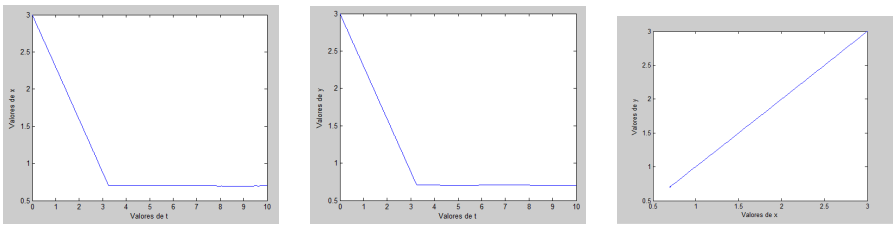


$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.13: Algoritmo Euler (función distancia)

Si ahora se cambian las coordenadas del punto inicial y se vuelve a resolver:

Parámetro	Valor	Parámetro	Valor
a	0	lambda	0.001
b	10	x0	3
m	1000	y0	3



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.14: Algoritmo Euler (función distancia)

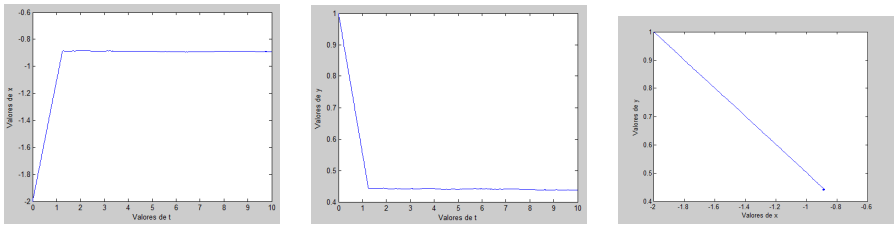
En ambos casos se observa cómo a partir del punto inicial, la solución de (2.6) es una recta perpendicular a la frontera de C (la mejor aproximación posible) que en un instante determinado alcanza el conjunto y a partir de entonces se mantiene constante. Ahora se va a incluir una perturbación de la forma

$$\mathbf{F}(x,y) = \begin{cases} (x,y) & \text{si } x,y \geq 0,\ x^2+y^2 \leq 4 \\ (0,0) & \text{en otro caso} \end{cases}$$

Se observa que el campo $\mathbf{F}$ actúa sólo en el primer cuadrante, por lo que no debería afectar al primer caso donde el punto inicial era $(-2,1)$ y sí al segundo

donde el punto inicial es $(3, 3)$. Vamos a comprobarlo al resolver el problema de nuevo mediante el método de Euler para ambos casos.

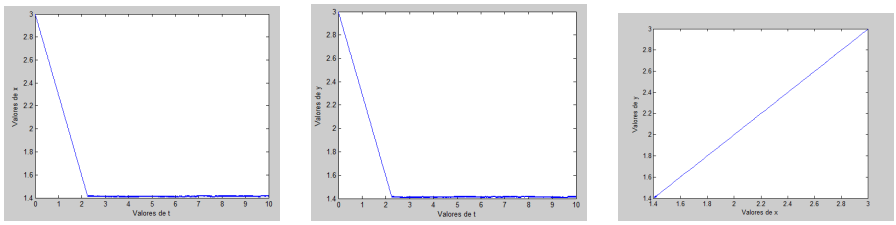
En el primer caso (manteniendo el resto de parámetros en la simulación) observamos que, en efecto, la solución se comporta de la misma forma:



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.15: Algoritmo Euler (función distancia+$\mathbf{F}$)

El segundo caso, por el contrario, se observa que debido al efecto del término fuente no se alcanza el conjunto C:



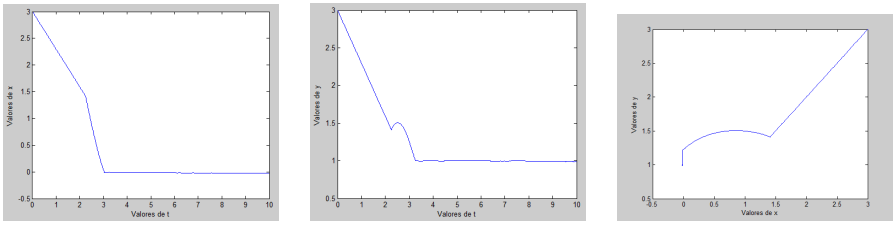
$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.16: Algoritmo Euler (función distancia+$\mathbf{F}$)

Finalmente, modificamos el término fuente, de forma que ahora la fuerza que se ejerce en el primer cuadrante el perpendicular a la trayectoria en el caso homogéneo

$$\mathbf{G}(x, y) = \begin{cases} (-y, x) & \text{si } x, y \geq 0,\ x^2 + y^2 \leq 4 \\ (0, 0) & \text{en otro caso} \end{cases}$$

Repetimos la simulación para el punto inicial $(3, 3)$ y observamos cómo el comportamiento de la solución cambia sustancialmente



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.17: Algoritmo Euler (función distancia+$\mathbf{G}$)

2.3 Caso bipotencial

Otra clase de problemas asociados a gradientes son los denominados *bipotenciales*, que tienen la forma

$$\left.\begin{aligned} -\dot{\mathbf{x}}(t) &\in \partial\phi(t,\mathbf{x}(t)) + \beta(t,\mathbf{x}(t))\partial\varphi(t,\mathbf{x}(t)) \\ \mathbf{x}(0) &= \mathbf{x}^0 \end{aligned}\right\} \qquad (\mathcal{P})$$

donde $\phi, \varphi : [0,T] \times \mathbb{R}^N \longrightarrow \mathbb{R}$ son dos funciones convexas en la segunda componente y $\beta : [0,T] \times \mathbb{R}^N \longrightarrow \mathbb{R}^N$ es un campo continuo. Tomando constantes $\lambda, \mu > 0$ y usando la regularización de Yosida de ambas subdiferenciales, queda el problema aproximado

$$\left.\begin{aligned} -\dot{\mathbf{x}}_{\lambda,\mu}(t) &= \mathcal{R}^{\phi}_{\lambda}(t,\mathbf{x}_{\lambda,\mu}(t)) + \beta(t,\mathbf{x}_{\lambda,\mu}(t))\mathcal{R}^{\varphi}_{\mu}(t,\mathbf{x}_{\lambda,\mu}(t)) \\ \mathbf{x}_{\lambda,\mu}(0) &= \mathbf{x}^0 \end{aligned}\right\} \qquad (\mathcal{P}_{\lambda,\mu})$$

donde

$$\mathcal{R}^{\phi}_{\lambda}(t,\mathbf{x}) = \frac{1}{\lambda}\left(\mathbf{x} - J^{\phi}_{\lambda}(t,\mathbf{x})\right), \qquad \mathcal{R}^{\varphi}_{\mu}(t,\mathbf{x}) = \frac{1}{\mu}\left(\mathbf{x} - J^{\varphi}_{\mu}(t,\mathbf{x})\right)$$

y J^{ϕ}_{λ}, J^{φ}_{μ} son las resolventes asociadas a los potenciales con diferentes niveles de aproximación

$$J^{\phi}_{\lambda}(t,\mathbf{x}) = \underset{\mathbf{y}\in\mathbb{R}^N}{\operatorname{argmin}}\left(\phi(t,\mathbf{x}) + \frac{1}{2\lambda}|\mathbf{y}-\mathbf{x}|^2\right)$$

$$J^{\varphi}_{\mu}(t,\mathbf{x}) = \underset{\mathbf{y}\in\mathbb{R}^N}{\operatorname{argmin}}\left(\varphi(t,\mathbf{x}) + \frac{1}{2\mu}|\mathbf{y}-\mathbf{x}|^2\right)$$

Aplicando los métodos de Euler y RK4 sobre $(\mathcal{P}_{\lambda,\mu})$ se obtendrán los problemas discretizados, cuyos algoritmos se describen a continuación.

2.3.1 Método de Euler

En este caso, para obtener el problema regularizado, habrá que añadir un nuevo paso al algoritmo de Euler para calcular la resolvente de Moreau asociada al

segundo potencial φ. También se modificará la actualización de los nodos en la discretización.

Modificación Euler (problemas bipotenciales)

2'. Cálculo de la segunda resolvente:

$$\mathbf{z}_{\mu}^{m,j} = \underset{\mathbf{z}\in\mathbb{R}^N}{\operatorname{argmin}} \left(\varphi(t^{m,j}, \mathbf{z}) + \frac{1}{2\mu} \left| \mathbf{z} - \mathbf{x}_{\lambda,\mu}^{m,j} \right|^2 \right)$$

3. Definición del nuevo nodo:

$$\mathbf{x}_{\lambda,\mu}^{m,j} = \mathbf{x}_{\lambda,\mu}^{m,j} - \frac{h_m}{\lambda} \left(\mathbf{x}_{\lambda,\mu}^{m,j} - \mathbf{y}_{\lambda}^{m,j} \right) - \beta(t^{m,j}, \mathbf{x}_{\lambda,\mu}^{m,j}) \frac{h_m}{\mu} \left(\mathbf{x}_{\lambda,\mu}^{m,j} - \mathbf{z}_{\mu}^{m,j} \right)$$

En este algoritmo se están suponiendo niveles de aproximación distintos para las regularizaciones de Yosida de las subdiferenciales de los potenciales, $\lambda > 0$ para ϕ y $\mu > 0$ para φ, de ahí el doble subíndice para indicar esta dependencia (en la inicialización se tomaría $\mathbf{x}_{\lambda,\mu}^{m,0} = \mathbf{x}_0$).

Al igual que antes, la solución de este problema mediante el método de Euler está programada en la función **euler.m**, y basta con introducir en ésta los argumentos de entrada adecuados para obtener una correcta resolución. En este caso éstos son los mismos que en el caso básico añadiendo `fun_prueba_2`, `mu` y `beta`. Como en ocasiones anteriores, la variación en el código se encuentra en el interior del bucle *for*. Aparece ahora en cada iteración, tras todo lo calculado en el caso básico, el valor proporcionado por la función `beta`, que se corresponde al valor de β evaluado en el paso de tiempo y en el valor del nodo correspondiente, multiplicando a una nueva regularización de Yosida correspondiente a la segunda función potencial. Para calcular esta resolvente se usa de nuevo la función **moreau.m** con su correspondiente uso de `fminsearch`. En este caso los argumentos de entrada de **moreau.m** son `x`, `x0`, `mu` y `fun_prueba_2`.

Como siempre, una vez que se ha terminado de recorrer el bucle *for*, la función devuelve el vector o la matriz con los valores de la solución para cada instante de tiempo.

2.3.2 Método de RK4

La modificación del algoritmo es algo complicada, ya que aparte de introducir un nuevo proceso de minimización para calcular la envoltura de Moreau del segundo potencial φ, hay que tener en cuenta los niveles de aproximación diferentes para las subdiferenciales de los potenciales.

Modificación RK4 (problemas bipotenciales)

- $\lambda, \mu > 0,\ m \geq 1$

  1. Inicialización: $\mathbf{x}_{\lambda,\mu}^{m,0} = \mathbf{x}^0$

  $$h_m = T/m$$

  $$j = 0$$

  2. Cálculo de los coeficientes:

  $$\mathbf{k}_{\lambda,\mu,1}^{m,j} = \frac{1}{\lambda}\left(\mathbf{x}_{\lambda,\mu}^{m,j} - J_\lambda^\phi(t^{m,j}, \mathbf{x}_{\lambda,\mu}^{m,j})\right)$$

  $$\mathbf{q}_{\lambda,\mu,1}^{m,j} = \frac{1}{\mu}\left(\mathbf{x}_{\lambda,\mu}^{m,j} - J_\mu^\varphi(t^{m,j}, \mathbf{x}_{\lambda,\mu}^{m,j})\right)$$

  $$\mathbf{p}_{\lambda,\mu,1}^{m,j} = -\mathbf{k}_{\lambda,\mu,1}^{m,j} - \beta(t^{m,j}, \mathbf{x}_{\lambda,\mu}^{m,j})\mathbf{q}_{\lambda,\mu,1}^{m,j}$$

  $$\mathbf{k}_{\lambda,\mu,2}^{m,j} = \frac{1}{\lambda}\left(\mathbf{x}_{\lambda,\mu}^{m,j} + \frac{h_m}{2}\mathbf{p}_{\lambda,\mu,1}^{m,j} - J_\lambda^\phi(t^{m,j} + \frac{h_m}{2}, \mathbf{x}_{\lambda,\mu}^{m,j} + \frac{h_m}{2}\mathbf{p}_{\lambda,\mu,1}^{m,j})\right)$$

  $$\mathbf{q}_{\lambda,\mu,2}^{m,j} = \frac{1}{\mu}\left(\mathbf{x}_{\lambda,\mu}^{m,j} + \frac{h_m}{2}\mathbf{p}_{\lambda,\mu,1}^{m,j} - J_\mu^\varphi(t^{m,j} + \frac{h_m}{2}, \mathbf{x}_{\lambda,\mu}^{m,j} + \frac{h_m}{2}\mathbf{p}_{\lambda,\mu,1}^{m,j})\right)$$

  $$\mathbf{p}_{\lambda,\mu,2}^{m,j} = -\mathbf{k}_{\lambda,\mu,2}^{m,j} - \beta(t^{m,j} + \frac{h_m}{2}, \mathbf{x}_{\lambda,\mu}^{m,j} + \frac{h_m}{2}\mathbf{p}_{\lambda,\mu,1}^{m,j})\mathbf{q}_{\lambda,\mu,2}^{m,j}$$

  $$\mathbf{k}_{\lambda,\mu,3}^{m,j} = \frac{1}{\lambda}\left(\mathbf{x}_{\lambda,\mu}^{m,j} + \frac{h_m}{2}\mathbf{p}_{\lambda,\mu,2}^{m,j} - J_\lambda^\phi(t^{m,j} + \frac{h_m}{2}, \mathbf{x}_{\lambda,\mu}^{m,j} + \frac{h_m}{2}\mathbf{p}_{\lambda,\mu,2}^{m,j})\right)$$

  $$\mathbf{q}_{\lambda,\mu,3}^{m,j} = \frac{1}{\mu}\left(\mathbf{x}_{\lambda,\mu}^{m,j} + \frac{h_m}{2}\mathbf{p}_{\lambda,\mu,2}^{m,j} - J_\mu^\varphi(t^{m,j} + \frac{h_m}{2}, \mathbf{x}_{\lambda,\mu,2}^{m,j} + \frac{h_m}{2}\mathbf{p}_{\lambda,\mu,2}^{m,j})\right)$$

  $$\mathbf{p}_{\lambda,\mu,3}^{m,j} = -\mathbf{k}_{\lambda,\mu,3}^{m,j} - \beta(t^{m,j} + \frac{h_m}{2}, \mathbf{x}_{\lambda,\mu}^{m,j} + \frac{h_m}{2}\mathbf{p}_{\lambda,\mu,2}^{m,j})\mathbf{q}_{\lambda,\mu,3}^{m,j}$$

  $$\mathbf{k}_{\lambda,\mu,4}^{m,j} = \frac{1}{\lambda}\left(\mathbf{x}_{\lambda,\mu}^{m,j} + h_m\mathbf{p}_{\lambda,\mu,3}^{m,j} - J_\lambda^\phi(t^{m,j} + h_m, \mathbf{x}_{\lambda,\mu}^{m,j} + h_m\mathbf{p}_{\lambda,\mu,3}^{m,j})\right)$$

  $$\mathbf{q}_{\lambda,\mu,4}^{m,j} = \frac{1}{\mu}\left(\mathbf{x}_{\lambda,\mu}^{m,j} + h_m\mathbf{p}_{\lambda,\mu,3}^{m,j} - J_\mu^\varphi(t^{m.j} + h_m, \mathbf{x}_{\lambda,\mu,3}^{m,j} + h_m\mathbf{p}_{\lambda,\mu,3}^{m,j})\right)$$

  $$\mathbf{p}_{\lambda,\mu,4}^{m,j} = -\mathbf{k}_{\lambda,\mu,4}^{m,j} - \beta(t^{m,j} + h_m, \mathbf{x}_{\lambda,\mu}^{m,j} + h_m\mathbf{p}_{\lambda,\mu,3}^{m,j})\mathbf{q}_{\lambda,\mu,4}^{m,j}$$

  para el cálculo de J_λ^ϕ, J_μ^φ ver página 62.

  3. Definición del nuevo nodo:

  $$\mathbf{x}_{\lambda,\mu}^{m,j+1} = \mathbf{x}_{\lambda.\mu}^{m,j} + \frac{h_m}{6}\left(\mathbf{p}_{\lambda,\mu,1}^{m,j} + 2\mathbf{p}_{\lambda,\mu,2}^{m,j} + 2\mathbf{p}_{\lambda,\mu,3}^{m,j} + \mathbf{p}_{\lambda,\mu,4}^{m,j}\right)$$

Al igual que con el método de Euler, basta con utilizar los argumentos de entrada adecuados para **rk4.m** para obtener una solución aproximada del problema. Los argumentos de entrada a utilizar son los mismos que en el método de Euler. La modificación del código aparece una vez más dentro del bucle *for*. A la hora de calcular los coeficientes `k1`, `k2`, `k3`, y `k4` correspondientes a los coeficientes $\mathbf{q}_{\lambda,\mu,1}^{m,j}$, $\mathbf{q}_{\lambda,\mu,2}^{m,j}$, $\mathbf{q}_{\lambda,\mu,3}^{m,j}$ y $\mathbf{q}_{\lambda,\mu,4}^{m,j}$ definidos en el psedocódigo hay que añadir respecto al caso básico lo mismo que se ha indicado en el método de Euler, es decir, el valor proporcionado por la función `beta` multiplicando a la regularización de Yosida de φ resuelta mediante **moreau.m**.

2.3.3 Ejemplos de aplicación: caso bipotencial

Presentamos en este apartado distintos experimentos numéricos en los que se corroboran los resultados teóricos del artículo de H. Attouch y M.-C. Czarnecki [1] sobre comportamiento asintótico de las soluciones

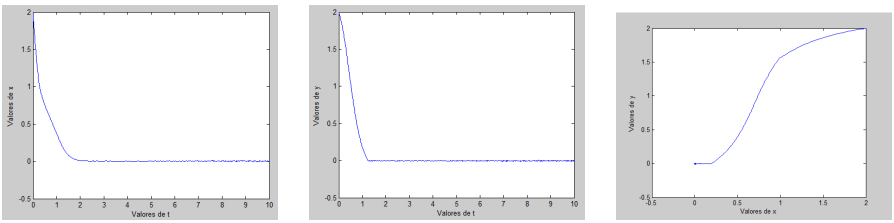
Ejemplo 2.5 Sean $I = [a_1, b_1]$, $J = [a_2, b_2] \subset \mathbb{R}$, dos intervalos compactos. Se consideran los potenciales convexos

$$\phi(x, y) = d_I(x) + d_J(y) + x^2 \tag{2.9}$$

$\varphi(x, y) = \frac{1}{2}(a_3 x - b_3 y)^2$ y la función $\beta(t) = t$, que verifican las condiciones del apartado 6.1 de [1].

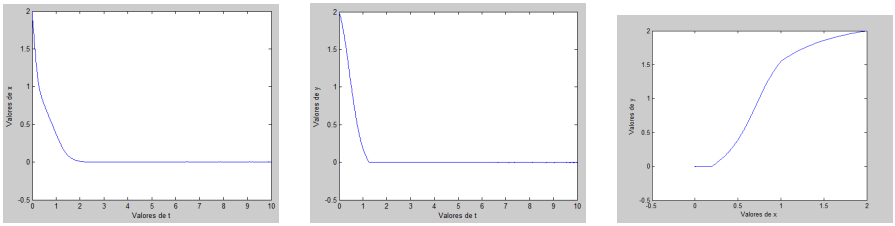
Se ejecutan los códigos Matlab para aproximar la solución del problema para estas funciones concretas y con los valores de los parámetros indicados en la tabla.

Parámetro	Valor	Parámetro	Valor	Parámetro	Valor
a	0	x0	2	b2	0
b	10	y0	2	a3	1
m	1000	a1	0	b3	2
lambda1	0.001	b1	1		
lambda2	0.001	a2	-1		



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.18: Euler, $\lambda = \mu = 0.001$

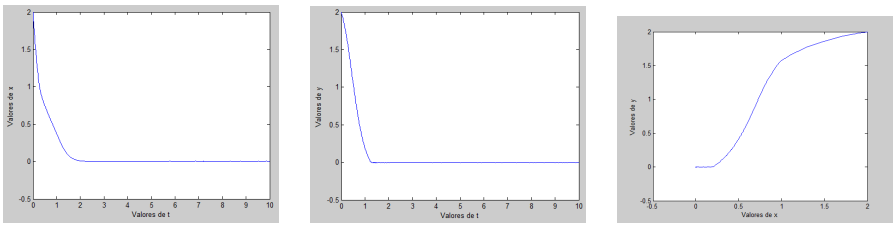


$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.19: Runge-Kutta, $\lambda = \mu = 0.001$

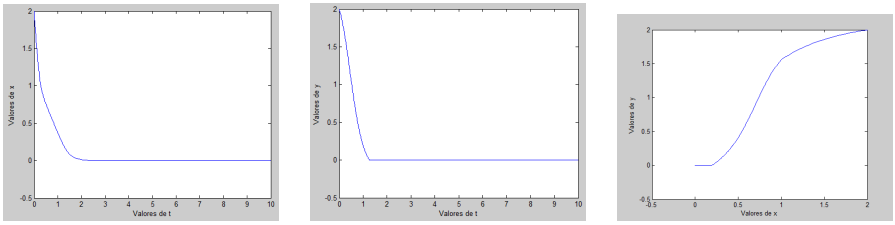
Del artículo citado sabemos que la solución de este problema, independientemente del punto inicial, converge a $(0, 0)$ cuando $t \to +\infty$. En nuestras simulaciones, para $t = 10$ se obtiene el valor $(0.0008, -0.0034)$ para el algoritmo de Euler y $(-0.0002, -0.0026)$ para el de Runge-Kutta.

Con objeto de valorar la sensibilidad de los algoritmos respecto del nivel de aproximación de las regularizaciones de Yosida de cada subdiferencial, repetimos las simulaciones anteriores manteniendo el valor de λ (`lambda1=0.001`) y modificando el de μ (`lambda2=0.01`). Se observa que aunque la forma de las gráficas es similar, el valor final varía notablemente respecto de la simulación anterior: $(0.0008, -0.0011)$ para Euler y $(0.0003, -0.0002)$ para Runge-Kutta.



$x(t)$ $y(t)$ $(x(t), y(t))$

Euler

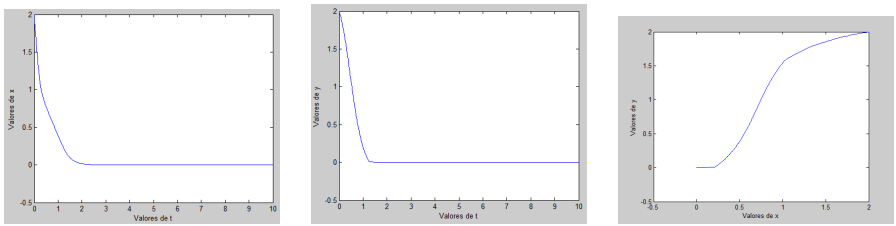


$x(t)$ $y(t)$ $(x(t), y(t))$

Runge-Kutta

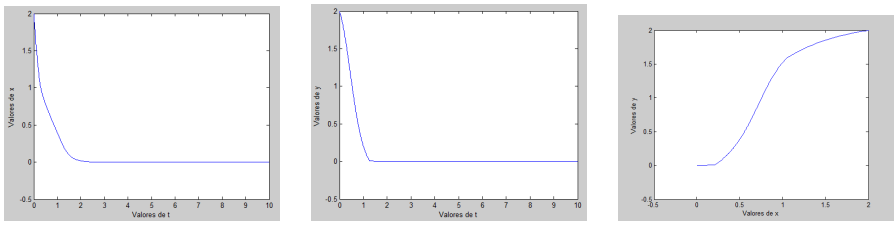
Figura 2.20: Simulaciones, $\lambda = 0.001,\ \mu = 0.01$

Finalmente, tomando los valores $\lambda = 0.0014$, $\mu = 0.013$, para $t = 10$ el algoritmo de Euler proporciona el valor $(0.0008, 0.0002)$ y el RK-4 $(0.0000, -0.0002)$.



$x(t)$ $y(t)$ $(x(t), y(t))$

Euler



$x(t)$ $y(t)$ $(x(t), y(t))$

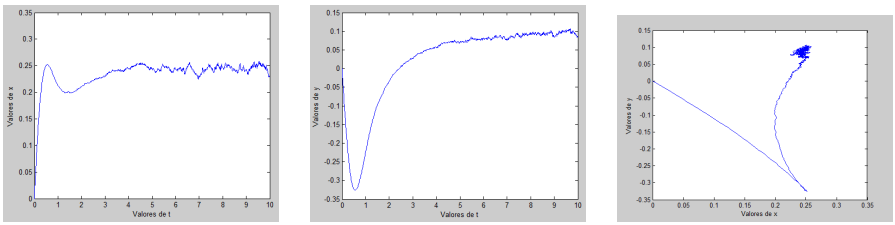
Runge-Kutta

Figura 2.21: Simulaciones, $\lambda = 0.014$, $\mu = 0.013$

Ejemplo 2.6 Vamos a considerar ahora el mismo problema que en el ejemplo anterior, cambiando los extremos de los intervalos I, J. Los valores de los distintos parámetros se recopilan en la siguiente tabla.

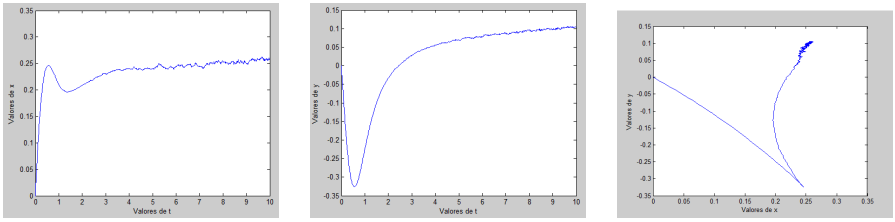
Parámetro	Valor	Parámetro	Valor	Parámetro	Valor
a	0	x0	0	a2	-1
b	10	y0	0	b2	-0.5
m	5000	a1	0.5	a3	1
lambda1	0.001	b1	1	b3	2
lambda2	0.001				

Sabemos, [1, Ecuación (23)], que cuando $t \to +\infty$ la solución del problema debe converger a $(0.25, 0.1)$. Nuestras simulaciones proporcionan para $t = 10$ los valores $(0.2308, 0.0861)$ (Euler) y $(0.2563, 0.1038)$ (Runge-Kutta). A continuación se muestran las gráficas obtenidas.



$x(t)$ $y(t)$ $(x(t), y(t))$

Euler



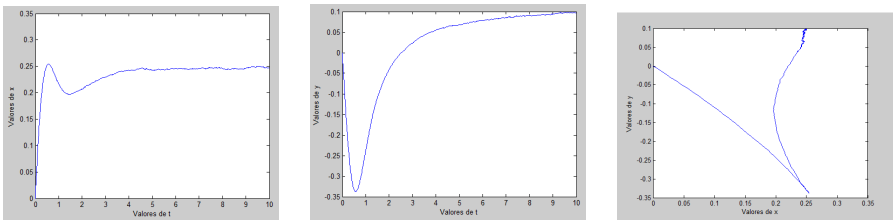
$x(t)$ $y(t)$ $(x(t), y(t))$

RK4

Figura 2.22: Ejemplo 2.6, $\lambda = \mu = 0.001$

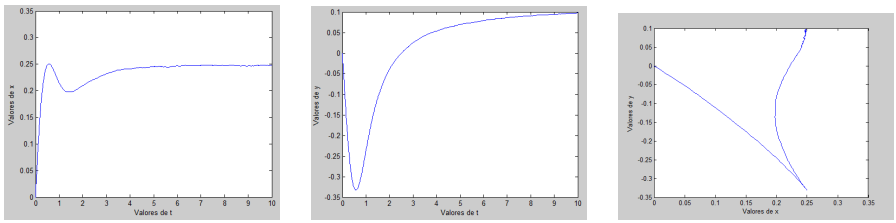
Realizamos nuevas simulaciones cambiando únicamente el nivel de aproximación de la regularización de Yosida de la segunda subdiferencial y para $t = 10$ se obtienen los valores de la solución $(0.2479, 0.0970)$ usando el método de Euler y $(0.2484, 0.0978)$ con Runge-Kutta, cuando $\mu = 0.01$. Finalmente, tomando $\mu = 0.011$, se tiene como valor aproximado en $t = 10$, $(0.2504, 0.0995)$ para Euler y $(0.2474, 0.0972)$ para Runge-Kutta.

Estas son las gráficas obtenidas:



$x(t)$ $y(t)$ $(x(t), y(t))$

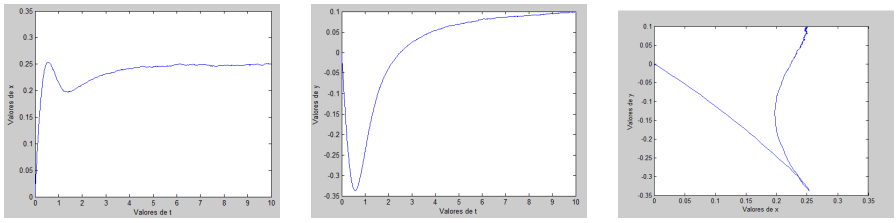
Euler



$x(t)$ $y(t)$ $(x(t), y(t))$

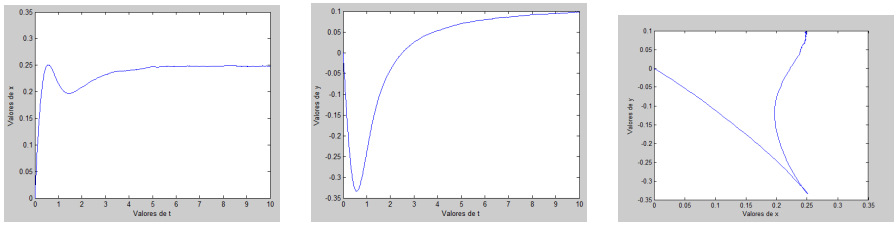
RK4

Figura 2.23: Ejemplo 2.6, $\lambda = 0.001$, $\mu = 0.01$



$x(t)$ $y(t)$ $(x(t), y(t))$

Euler



$x(t)$ $y(t)$ $(x(t), y(t))$

RK4

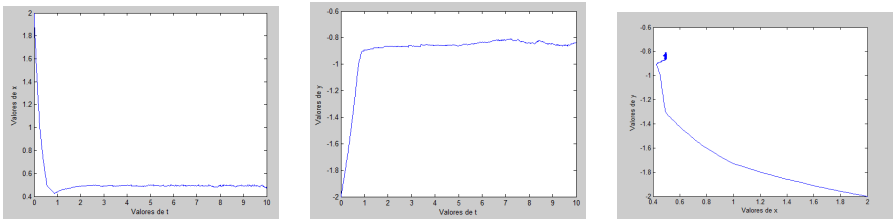
Figura 2.24: Ejemplo 2.6, $\lambda = 0.001$, $\mu = 0.011$

Ejemplo 2.7 Para finalizar se consideran las mismas funciones ϕ y β que en el Ejemplo 2.6 y se toma φ como la función distancia al disco unidad,

$$\mathcal{B} = \left\{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\right\}$$

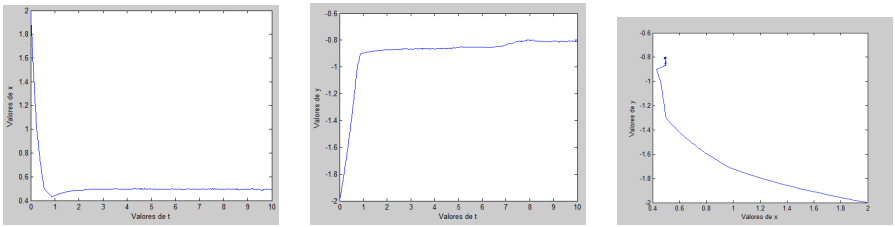
cuya expresión se ha usado en ejemplos anteriores. El valor de los distintos parámetros viene dado por la tabla.

Parámetro	Valor	Parámetro	Valor
a	0	x0	2
b	10	y0	-2
m	10000	a1	0.5
lambda1	0.001	b1	1
lambda2	0.001	a2	-1
		b2	-0.5



$x(t)$ $y(t)$ $(x(t), y(t))$

Euler



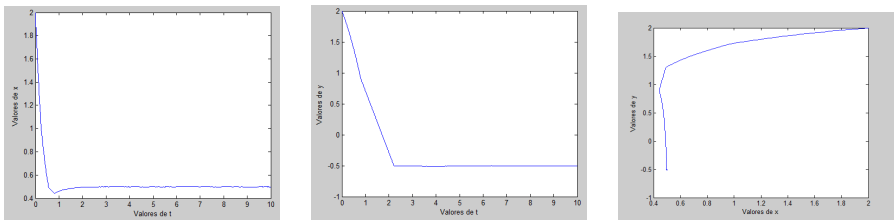
$x(t)$ $y(t)$ $(x(t), y(t))$

RK4

Figura 2.25: Ejemplo 2.7, $\lambda = \mu = 0.001$

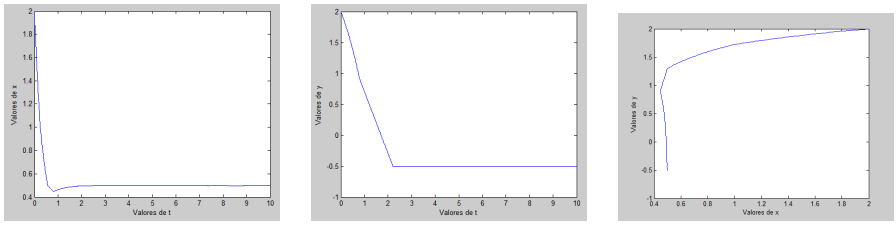
Del Th. 3.1 en [1] (ver Teorema 1.18 en el Capítulo 1), se sigue que cuando $t \to +\infty$, la solución del problema converge a un punto en $\operatorname{argmin}_{\mathcal{B}} \phi = \mathcal{B} \cap (\{0.5\} \times [-1, -0.5])$. Las simulaciones están de acuerdo con este resultado ya que para $t = 10$, obtenemos los valores $(0.4801, -0.8328)$ (Euler) y $(0.4954, -0.8112)$ (Runge-Kutta), que nos permiten conjeturar que la solución converge asintóticamente a al punto $(0.5, -0.8) \in \operatorname{argmin}_{\mathcal{B}} \phi$.

Finalmente simulamos la solución a partir del punto inicial $(2, 2)$, tomando todos los parámetros iguales a excepción de $\mu = 0.01$.



$x(t)$ $y(t)$ $(x(t), y(t))$

Euler



$x(t)$ $y(t)$ $(x(t), y(t))$

RK4

Figura 2.26: Ejemplo 2.7, $y_0 = 2$, $\mu = 0.01$

Respecto al comportamiento asintótico, para $t = 10$ los algoritmos porporcionan los valores $(0.4974, -0.5065)$ (Euler) y $(0.4997, -0.5016)$ (Runge-Kutta), lo que nos permite conjeturar que la solución tenderá hacia $(0.5, -0.5) \in \operatorname{argmin}_{\mathcal{B}} \phi$.

2.4 Caso bipotencial con término fuente

Si se combinan los problemas de las dos secciones anteriores se obtiene:

$$\left.\begin{aligned} -\dot{\mathbf{x}}(t) &\in \partial\phi(t, \mathbf{x}(t)) + \beta(t, \mathbf{x}(t))\partial\varphi(t, \mathbf{x}(t)) - \mathbf{F}(t, \mathbf{x}(t)) \\ \mathbf{x}(0) &= \mathbf{x}^0 \end{aligned}\right\} \qquad (\mathcal{P})$$

Tomando $\lambda, \mu > 0$ y manteniendo la notación del apartado anterior se tiene el problema aproximado $(\mathcal{P}_{\lambda,\mu})$

$$\left.\begin{aligned} -\dot{\mathbf{x}}_{\lambda,\mu}(t) &= \mathcal{R}^{\phi}_{\lambda}(t, \mathbf{x}_{\lambda,\mu}(t)) + \beta(t, \mathbf{x}_{\lambda,\mu}(t))\mathcal{R}^{\varphi}_{\mu}(t, \mathbf{x}_{\lambda,\mu}(t)) - \mathbf{F}(t, \mathbf{x}_{\lambda}(t)) \\ \mathbf{x}_{\lambda,\mu}(0) &= \mathbf{x}^0 \end{aligned}\right\}$$

Seguidamente veremos cómo hay que modificar los algoritmos del apartado anterior para incorporar el término fuente.

2.4.1 Método de Euler

En este caso solamente hay que modificar el tercer paso del algoritmo de Euler para el problema bipotencial incorporando el campo $\mathbf{F}$ a la hora de definir el nuevo nodo .

Modificación Euler (problemas bipotenciales con término fuente)

---

3. Definición del nuevo nodo:

$$\mathbf{x}_{\lambda,\mu}^{m,j} = \mathbf{x}_{\lambda,\mu}^{m,j} - \frac{h_m}{\lambda}\left(\mathbf{x}_{\lambda,\mu}^{m,j} - \mathbf{y}_{\lambda}^{m,j}\right) \quad - \quad \beta(t^{m,j}, \mathbf{x}_{\lambda,\mu}^{m,j})\frac{h_m}{\mu}\left(\mathbf{x}_{\lambda,\mu}^{m,j} - \mathbf{z}_{\mu}^{m,j}\right)$$
$$+ \quad h_m \mathbf{F}(t^{m,j}, \mathbf{x}_{\lambda,\mu}^{m,j})$$

Los argumentos de entrada a introducir en **euler.m** son ahora la combinación de los del *caso bipotencial* y el *caso con término fuente*, es decir, `fun_prueba`, `x0`, `a`, `b`, `m`, `lambda`, `term_fuente`, `fun_prueba_2`, `mu` y `beta`. Todos ellos ya están explicados con anterioridad por lo que no se volverán a repetir. Además, los cambios a realizar en el código no son más que la inclusión de los términos del *caso bipotencial* y los del *caso con término fuente*, por lo que tampoco se repetirá su explicación.

2.4.2 Método de RK4

Al igual que en el método de Euler, el algoritmo de Runge-Kutta se obtiene combinando los anteriores. Sin embargo en ese caso la expresión es más complicada dado que el término fuente interviene en el cálculo de los cuatro coeficientes que hay que actualizar en cada iteración. Por tanto, manteniendo la notación del apartado anterior, se tiene

---

Modificación RK4 (bipotencial con término fuente)

---

2. Cálculo de los coeficientes:

$$\begin{aligned}
\mathbf{p}_{\lambda,\mu,1}^{m,j} &= -\mathbf{k}_{\lambda,\mu,1}^{m,j} - \beta(t^{m,j}, \mathbf{x}_{\lambda,\mu}^{m,j})\mathbf{q}_{\lambda,\mu,1}^{m,j} + \mathbf{F}(t^{m,j}, \mathbf{x}_{\lambda,\mu}^{m,j}) \\
\mathbf{p}_{\lambda,\mu,2}^{m,j} &= -\mathbf{k}_{\lambda,\mu,2}^{m,j} - \beta(t^{m,j} + \tfrac{h_m}{2}, \mathbf{x}_{\lambda,\mu}^{m,j} + \tfrac{h_m}{2}\mathbf{p}_{\lambda,\mu,1}^{m,j})\mathbf{q}_{\lambda,\mu,2}^{m,j} \\
&\quad + \mathbf{F}(t^{m,j} + \tfrac{h_m}{2}, \mathbf{x}_{\lambda,\mu}^{m,j} + \tfrac{h_m}{2}\mathbf{p}_{\lambda,\mu,1}^{m,j}) \\
\mathbf{p}_{\lambda,\mu,3}^{m,j} &= -\mathbf{k}_{\lambda,\mu,3}^{m,j} - \beta(t^{m,j} + \tfrac{h_m}{2}, \mathbf{x}_{\lambda,\mu}^{m,j} + \tfrac{h_m}{2}\mathbf{p}_{\lambda,\mu,2}^{m,j})\mathbf{q}_{\lambda,\mu,3}^{m,j} \\
&\quad + \mathbf{F}(t^{m,j} + \tfrac{h_m}{2}, \mathbf{x}_{\lambda,\mu}^{m,j} + \tfrac{h_m}{2}\mathbf{p}_{\lambda,\mu,2}^{m,j})
\end{aligned}$$

$$\mathbf{p}_{\lambda,\mu,4}^{m,j} = -\mathbf{k}_{\lambda,\mu,4}^{m,j} - \beta(t^{m,j} + h_m, \mathbf{x}_{\lambda,\mu}^{m,j} + h_m \mathbf{p}_{\lambda,\mu,3}^{m,j})\mathbf{q}_{\lambda,\mu,4}^{m,j} + \mathbf{F}(t^{m,j} + h_m, \mathbf{x}_{\lambda,\mu}^{m,j} + h_m \mathbf{p}_{\lambda,\mu,3}^{m,j})$$

Los argumentos de entrada necesarios para la función **rk4.m** son los mismos que los del método de Euler para este problema y al igual que en éste, el código es una suma del *caso bipotencial* y el *caso con término fuente*.

2.4.3 Ejemplo de aplicación: sistema mecánico con fricción de Coulomb

• **Modelo físico.** Consideremos un sistema mecánico formado por una masa m suspendida de un muelle elástico de forma que oscila en el interior de un cilindro lleno de un fluido (en la Figura 2.27 se observa un dispositivo de esa naturaleza).

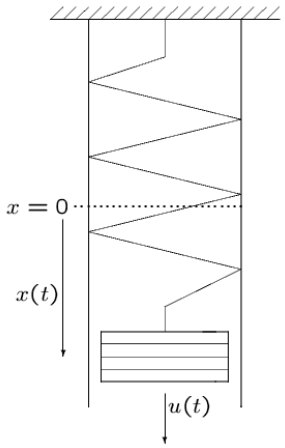


Figura 2.27: Dispositivo con fricción de Coulomb

Sea $x(t)$ la posición vertical de la masa a lo largo del tiempo. De la Ley fundamental de la dinámica sabemos que

$$m\ddot{x}(t) = F \tag{2.11}$$

donde F es la resultante de todas las fuerzas que actúan en el sistema. Suponiendo que el muelle es elástico, con $k > 0$ su constante de elasticidad, la ley de Hooke nos dice que su aportación al movimiento será de la forma

$$-kx(t) \tag{2.12}$$

Por su parte, el rozamiento con el fluido genera un término de viscosidad

$$-\eta\dot{x}(t) \tag{2.13}$$

mientras que la fricción del sólido con las paredes del cilindro produce un rozamiento de Coulomb (o fricción dry) que se describe mediante el término multivaluado

$$-\alpha\,\mathrm{Sgn}(\dot{x}(t)) \tag{2.14}$$

donde

$$\mathrm{Sgn}(z) = \begin{cases} 1, & z > 0 \\ [-1,1], & z = 0 \\ -1, & z < 0 \end{cases}$$

En realidad, esto correspondería a un modelo simplificado en que el rozamiento con las paredes genera la misma resistencia independientemente del sentido del desplazamiento. Pero en algunos casos, esto no es cierto y el término asociado a la fricción de Coulomb toma la forma

$$-\Gamma(\dot{x}(t)) \tag{2.15}$$

con

$$\Gamma(z) = \begin{cases} \alpha^+, & z > 0 \\ [-\alpha^-, \alpha^+], & z = 0 \\ -\alpha^-, & z < 0 \end{cases} \tag{2.16}$$

$\alpha^+, \alpha^- > 0$ constantes.

Finalmente se tendría la fuerza exterior $u(t)$, lo que nos da una resultante de fuerzas

$$F = u(t) - kx(t) - \eta\dot{x}(t) - \Gamma(\dot{x}(t)) \tag{2.17}$$

que junto con (2.11) permite escribir la inclusión de segundo orden

$$m\ddot{x}(t) + kx(t) + \eta\dot{x}(t) - u(t) \in -\Gamma(\dot{x}(t)) \tag{2.18}$$

Es sencillo comprobar que, si $\phi(z) = \max(\alpha^+ z, -\alpha^- z)$, entonces $\partial\phi = \Gamma$, lo que nos proporciona la forma subdiferencial de la ecuación que rige el estado del sistema

$$-m\ddot{x}(t) - kx(t) - \eta\dot{x}(t) + u(t) \in \partial\phi(\dot{x}(t)) \tag{2.19}$$

Si consideramos un sistema más complejo en que la masa m está unida a dos muelles en paralelo sumergidos ambos en sendos fluidos y de forma que existe rozamiento con las paredes (Figura 2.28), al ser la fuerza total la suma de las fuerzas se tiene que el desplazamiento transversal de la masa se describe mediante la ecuación

$$-m\ddot{x}(t) - kx(t) - \eta\dot{x}(t) + u(t) \in \partial\phi_1(\dot{x}(t)) + \partial\phi_2(\dot{x}(t)) \tag{2.20}$$

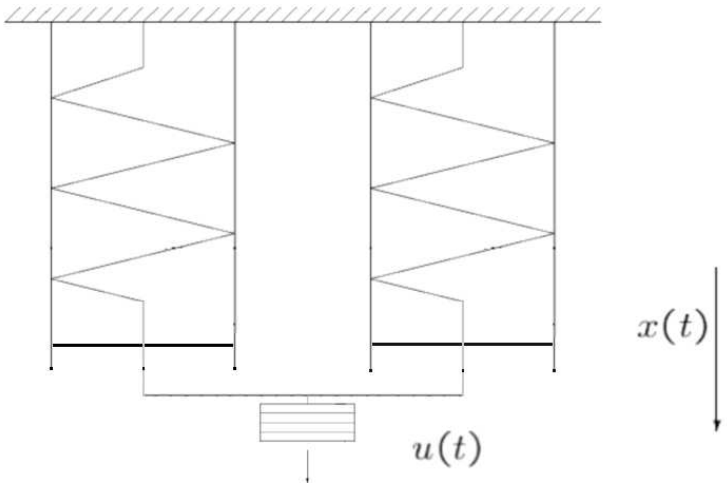


Figura 2.28: Dispositivos en paralelo

donde $k = k_1 + k_2$ es la suma de las constantes de rigidez de los muelles, $\eta = \eta_1 + \eta_2$ la de las viscosidades y α_j^+, α_j^-, $j = 1, 2$, son las constantes asociadas a la fricción seca, $\phi_j(z) = \max\left(\alpha_j^+ z,\, -\alpha_j^- z\right)$, $j = 1, 2$.

• **Modelo matemático.** Obviamente (2.20) puede escribirse como un sistema de la forma

$$\begin{cases} \dot{x}(t) = y(t) \\ -\dot{y}(t) \in \frac{1}{m}\left(kx(t) + \eta\dot{x}(t) - u(t)\right) + \partial\phi_1(y(t)) + \partial\phi_2(y(t)) \end{cases} \tag{2.21}$$

y, si definimos el término fuente

$$\mathbf{F}(t, x, y) = \left(y,\, \frac{-kx - \eta y + u(t)}{m}\right)$$

y los potenciales convexos $\Phi_j(x, y) = \phi_j(y)$, $j = 1, 2$, tenemos escrita la ecuación del sistema en la forma estándar de una inclusión diferencial de tipo bipotencial con término fuente:

$$-(\dot{x}(t), \dot{y}(t)) \in \partial\Phi_1(x(t), y(t)) + \partial\Phi_2(x(t), y(t)) - \mathbf{F}(t, x(t), y(t)) \tag{2.22}$$

Podemos, pues, usar los algoritmos que acabamos de exponer para simular el comportamiento de un sistema físico del tipo descrito. Cabe decir que en este caso el cálculo de la resolvente se simplifica, teniendo en cuenta que, para cada

$(x, y) \in \mathbb{R}^2$:

$$\operatorname*{argmin}_{(z_1,z_2)\in\mathbb{R}^2} \Phi_j(z_1, z_2) + \frac{1}{2\lambda}\left((z_1 - x)^2 + (z_2 - y)^2\right) =$$

$$\operatorname*{argmin}_{(z_1,z_2)\in\mathbb{R}^2} \phi_j(z_2) + \frac{1}{2\lambda}\left((z_1 - x)^2 + (z_2 - y)^2\right) =$$

$$\left(x, \operatorname*{argmin}_{z_2\in\mathbb{R}} \left(\phi_j(z_2) + \frac{1}{2\lambda}(z_2 - y)^2\right)\right)$$

• **Simulaciones numéricas.** Se van a resolver estas ecuaciones para varios casos distintos de fuerza exterior $u(t)$. Se usará solamente el método de Euler.

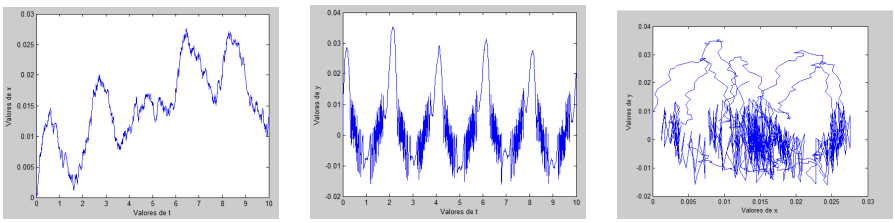
En primer lugar se supondrá una fuerza exterior periódica $u(t) = \cos(\pi t)$. Los parámetros del modelo físico considerados son

Parámetro	Valor	Parámetro	Valor
k_1	1	α_1^+	0.4
α_1^-	0.4	η_1	1
k_2	1	α_2^+	0.4
α_2^-	0.6	η_2	2
m	1	$u(t)$	$\cos(\pi t)$

Y los del método numérico usados en las simulaciones son:

Parámetro	Valor	Parámetro	Valor
`a`	`0`	`nu`	`0.001`
`b`	`10`	`beta`	`1`
`m`	`1000`	`x0`	`0`
`lambda`	`0.001`	`y0`	`0`

Con lo que se obtienen los siguientes gráficos para la posición $x(t)$ de la masa a lo largo del tiempo, su velocidad $y(t) = \dot{x}(t)$ y la curva $(x(t), y(t))$:



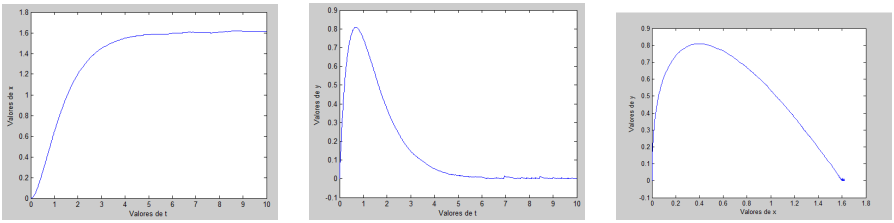
$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.29: Fuerza exterior $u(t) = \cos(\pi t)$

Si se cambia la fuerza exterior y se supone de valor constante, por ejemplo, $u(t) = 4$, los parámetros utilizados en el problema y los resultados obtenidos ahora son:

Parámetro	Valor	Parámetro	Valor
k_1	1	α_1^+	0.4
α_1^-	0.4	η_1	1
k_2	1	α_2^+	0.4
α_2^-	0.6	η_2	2
m	1	$u(t)$	4

Parámetro	Valor	Parámetro	Valor
`a`	`0`	`nu`	`0.001`
`b`	`10`	`beta`	`1`
`m`	`1000`	`x0`	`0`
`lambda`	`0.001`	`y0`	`0`



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.30: Fuerza exterior $u(t) = 4$

Por último se supondrá una fuerza exterior que no actúa durante todo el período de tiempo, sino sólo una parte del mismo. La expresión de la fuerza considerada ahora será:

$$u(t) = \left\{ \begin{array}{cc} \cos(\pi t) & \text{si } 0 < t < 2 \\ 0 & \text{en otro caso} \end{array} \right\} = \mathcal{X}_{[0,2]}(t) \cos(\pi t)$$

con

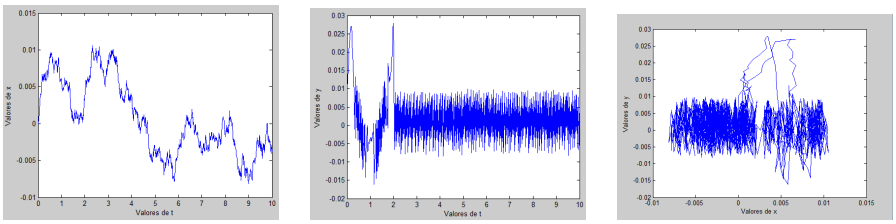
$$\mathcal{X}_{[0,2]}(t) = \left\{ \begin{array}{ll} 1, & \text{si } 0 \leq t \leq 2 \\ 0, & \text{en otro caso} \end{array} \right.$$

la función característica del intervalo $[0, 2]$.

Los parámetros utilizados y los resultados obtenidos son ahora

Parámetro	Valor	Parámetro	Valor
k_1	1	α_1^+	0.4
α_1^-	0.4	η_1	1
k_2	1	α_2^+	0.4
α_2^-	0.6	η_2	2
m	1	$u(t)$	$\mathcal{X}_{[0,2]}(t)\cos(\pi t)$

Parámetro	Valor	Parámetro	Valor
a	0	nu	0.001
b	10	beta	1
m	1000	x0	0
lambda	0.001	y0	0



$x(t)$ $y(t)$ $(x(t), y(t))$

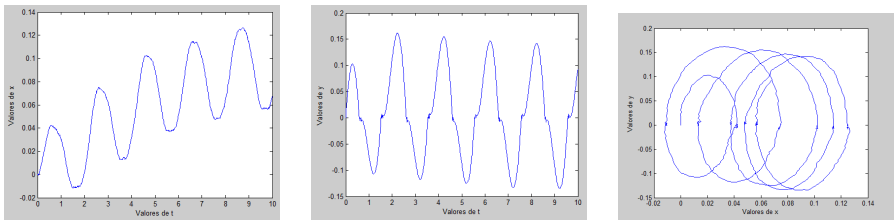
Figura 2.31: Fuerza exterior $u(t) = \mathcal{X}_{[0,2]}(t)\cos(\pi t)$

Se puede repetir alguna de las simulaciones realizadas suponiendo ahora que hay sólo un muelle en el sistema en lugar de dos. Por ejemplo, si se vuelve a estudiar el caso de una fuerza exterior periódica $u(t) = \cos(\pi t)$ en la que ahora los parámetros son:

Parámetro	Valor	Parámetro	Valor
k_1	0	α_1^+	0
α_1^-	0	η_1	0
k_2	1	α_2^+	0.4
α_2^-	0.6	η_2	2
m	1	$u(t)$	$\cos(\pi t)$

Parámetro	Valor	Parámetro	Valor
a	0	nu	0.001
b	10	beta	1
M	1000	x0	0
lambda	0.001	y0	0

se obtienen los siguientes resultados



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.32: Fuerza exterior $u(t) = \cos(\pi t)$, un muelle

Es importante reseñar que el software desarrollado en este proyecto permite simular el comportamiento de sistemas más complejos en los que interaccionan muelles elásticos, elementos de Newton (con rozamiento viscoso) y elementos de Saint-Venant (rozamiento seco) en serie y/o en paralelo.

Capítulo 3

Interfaz gráfica

Todos los casos que se han estudiado en el anterior capítulo han sido recogidos e implementados en una interfaz gráfica para facilitar el manejo de los códigos por parte del usuario. Esta interfaz permite introducir de manera sencilla y sin conocimientos profundos de programación todos los parámetros necesarios para la resolución de problemas asociados a diversos potenciales particulares. Permite desde definir el tipo de problema a resolver e introducir todos sus parámetros hasta exportar e importar resultados, realizar una animación con los mismos y otras diversas características que se irán detallando en el presente capítulo.

Para crear esta interfaz gráfica se ha utilizado la herramienta GUIDE, integrada en MATLAB. Con ella se pueden disponer en la pantalla los diferentes elementos (botones, cuadros de texto, menús, etc.) y definir sus atributos (tamaño, aspecto, color, etc.) de forma más sencilla a como resulta introduciéndolos directamente mediante código. Esto genera un archivo **GUI.fig** que contiene toda la información gráfica y el cual lleva asociado otro archivo **GUI.m** en el que se definen, ahora sí mediante código, las acciones a realizar por cada botón y cuadro de texto y se establecen las variables globales necesarias para el correcto funcionamiento de las funciones utilizadas. En este archivo ha sido en el que se ha realizado el mayor esfuerzo de programación y en el cual se ha invertido un mayor tiempo con diferencia, ya que se han debido cuidar bien todos los aspectos del programa y conseguir un correcto funcionamiento coordinando la gran cantidad de funciones usadas.

En la Figura 3.1 se presenta una captura de la pantalla principal del programa. A lo largo del capítulo se irán desarrollando cada una de las partes y características del mismo.

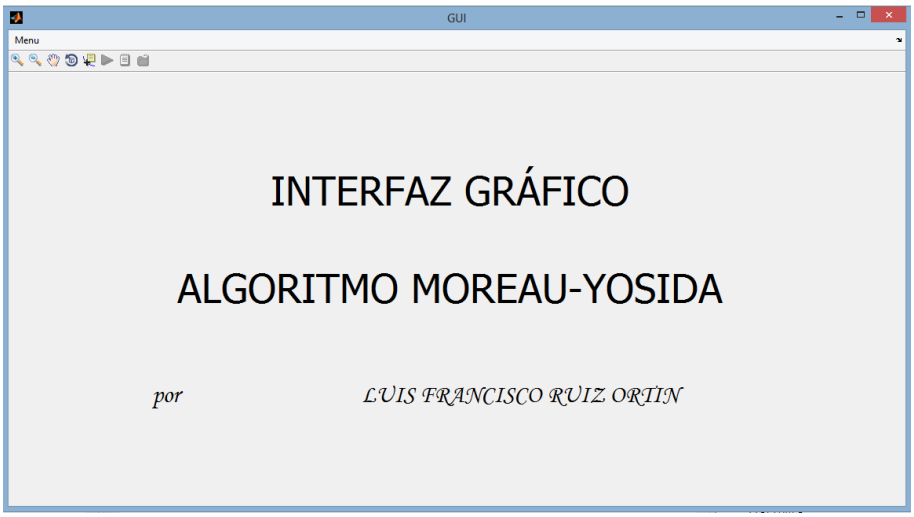


Figura 3.1: Pantalla principal del programa

3.1 Menú

Lo primero que aparece en la parte superior de la ventana es el menú. En él

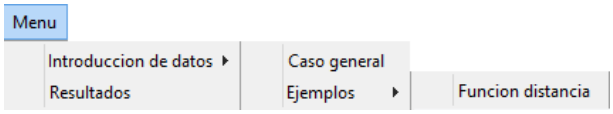


Figura 3.2: Menú desplegado

se elige entre acceder al panel de introducción de datos del problema o al panel de resultados. Dentro de la introducción de datos, se puede acceder al caso general que corresponde a todos los casos estudiados en el capítulo anterior o a un ejemplo concreto de aplicación donde la función potencial es la distancia a un conjunto. Este ejemplo con la función distancia será estudiado con más profundidad en el siguiente capítulo, por lo que ahora se abordará el únicamente el caso general.

3.2 Barra de herramientas

Lo siguiente que aparece en la ventana gráfica, justo debajo del menú, es la barra de herramientas.

Figura 3.3: Barra de herramientas

En ella se encuentran diversas herramientas que se pueden utilizar en el programa. A continuación se detalla la utilidad y cúando se puede utilizar cada una de ellas:

Acercar zoom: Esta herramienta permite acercar el zoom en los resultados gráficos del panel de resultados.

Alejar zoom: Esta herramienta permite alejar el zoom en los resultados gráficos del panel de resultados.

Mover: Esta herramienta permite moverse dentro de una ventana gráfica del panel de resultados.

Girar vista: Esta herramienta permite girar el punto de vista en los resultados gráficos del panel de resultados. Es especialmente útil para el caso de resultados en tres dimensiones.

Cursor: Esta herramienta muestra, al pulsar en un punto dentro de una gráfica de resultados, los valores de las diferentes variables representadas en esa gráfica en dicho punto.

Repetir: Esta herramienta permite, una vez que se ha completado el cálculo de la solución de un problema y su representación gráfica, repetir esta última. Esto es especialmente útil en el caso de que la representación gráfica sea una animación, ya que permite repetirla todas las veces que se desee. Esta herramienta se encuentra en principio desactivada y sólo se activará una vez que se haya realizado el cálculo de un determinado problema.

Exportar resultados: Esta herramienta permite, una vez que se ha realizado el cálculo de un problema y su representación gráfica, exportar los valores de la solución a un archivo *.txt*. Al pulsar sobre este botón te permite seleccionar la ubicación y el nombre del archivo *.txt* que se quiera generar. En él aparecerán, en columnas diferenciadas, cada una de las dimensiones de la solución (sus valores para cada instante de tiempo) , la

distancia de la solución al conjunto (sólo para problemas de persecución, que se presentarán más adelante), el vector de tiempos y las componentes necesarias para la representación del conjunto (sólo para problemas de persecución en los que se quiera visualizar una animación de los resultados). Esta herramienta se encuentra en principio desactivada y sólo se activará una vez que se haya realizado el cálculo de un determinado problema.

Cargar resultados: Esta herramienta permite cargar resultados calculados previamente y almacenados en un archivo *.txt* y representarlos gráficamente. Al pulsar este botón se abre una ventana que nos permite seleccionar el archivo *.txt* en el que se encuentran los datos. Una vez que se selecciona el archivo se abre una ventana que nos permite seleccionar el tipo de problema del que se trata y una vez seleccionado procede a su representación. Esta herramienta se encuentra en principio desactivada y sólo se activará cuando nos encontremos en el panel de resultados.

3.3 Panel de introducción de datos

El panel de introducción de datos se abre al seleccionarlo en el menú. Como se ha comentado antes, se va a tratar ahora sólo el panel de datos para el caso general, dejando la aplicación particular de la función distancia (problema de persecución) para otro capítulo más adelante.

El aspecto que presenta el panel de introducción de datos para el caso general es el siguiente:

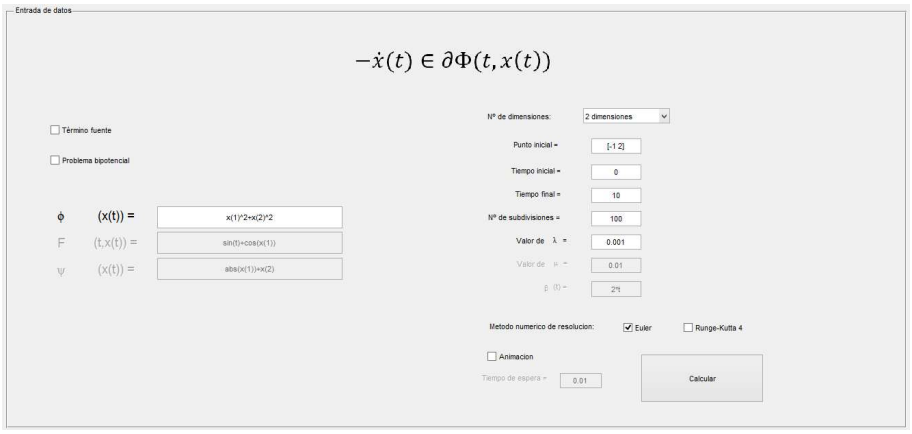


Figura 3.4: Panel de introducción de datos

Se pueden observar diversos elementos en él. En primer lugar, en la parte superior se encuentra la formulación matemática del problema ($\mathcal{P}$) consider-

ado. Esta formulación, como podemos observar en la Figura 3.5, varía según el problema considerado.

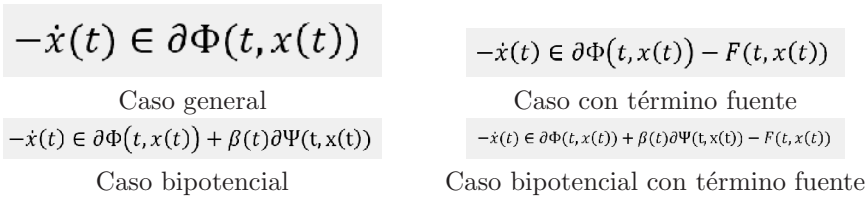


Figura 3.5: Formulación matemática del problema

Para seleccionar el tipo de problema a resolver se hace uso de la serie de botones que aparecen en la Figura 3.6.

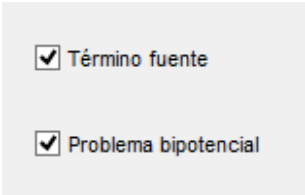


Figura 3.6: Botones de selección del tipo de problema

Aparecen también cuadros de texto donde se pueden escribir las expresiones de la o las funciones potencial a utilizar y el término fuente si éste apareciera en el problema, como podemos ver en la Figura 3.7. Esta información se corresponde a la introducida en las funciones `fun_prueba`, `fun_prueba_2` y `term_fuente` que han sido explicadas en capítulos anteriores. Estas funciones se han modificado para que tomen los datos introducidos en la interfaz gráfica (los cuales son de tipo `string`), los conviertan a funciones y devuelvan su valor evaluado en cada punto que se le ingresa. De esta manera, de manera externa al código se pueden variar las expresiones de las funciones de forma rápida y cómoda.

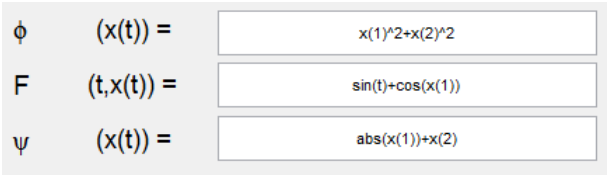


Figura 3.7: Ejemplos de funciones a utilizar

En la parte de la derecha aparecen el resto de parámetros a introducir: n de dimensiones (`dim`), `x0`, `a`, `b`, `m`, `lambda`, `nu` y `beta` (estos dos últimos parámetros están inactivos en pricipio y sólo se activan si se marca que se rata de un problema bipotencial). Esto se puede ver en la Figura 3.8.

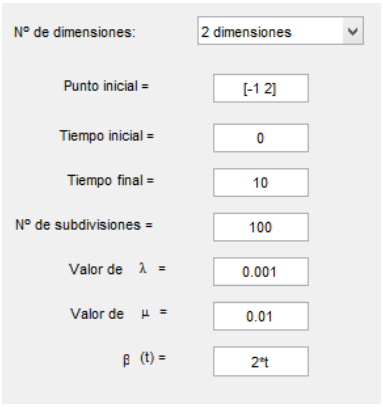


Figura 3.8: Introducción de parámetros

Por último se elige el método numérico a utilizar para la resolución del problema aproximado, que denotamos ($\mathcal{P}_\lambda$). Como se ha visto anteriormente, éste puede ser el método de Euler o el método de Runge-Kutta de orden 4. Si se marcan ambos en el panel de resultados se mostrará una comparación entre los resultados obtenidos con cada uno de ellos. Se puede seleccionar si se desea que las gráficas de la solución no se muestren de golpe, sino como una animación de su evolución a lo largo del tiempo. Para ello hay que marcar la opción de animación y establecer el tiempo de espera deseado entre un punto y el siguiente. Esto se muestra en la Figura 3.9.

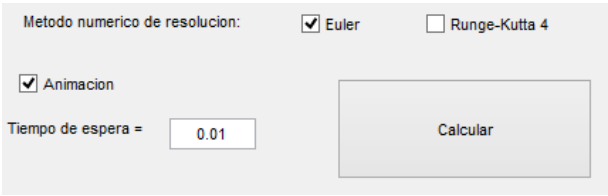


Figura 3.9: Opciones de resolución

Una vez que hemos definido todos los parámetros del problema a resolver se debe pulsar el botón *Calcular* para obtener la solución. Una vez que se presiona este botón, aparece una barra de progreso que muestra el avance del cálculo (Figura 4.7). Cuando haya terminado, el programa automáticamente pasará al panel de resultados y mostrará lo solución del problema.

3.4 Panel de resultados

En este panel se muestran las gráficas de los resultados obtenidos. Si se selecciona antes de haber calculado nada aparecerá en blanco como muestra la

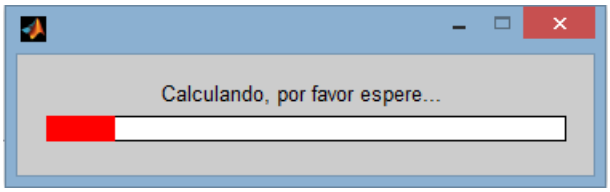


Figura 3.10: Progreso del cálculo

Figura 3.11, pero podrá usarse desde el principio para representar resultados calculados con anterioridad con la herramienta *Cargar datos*. Sin embargo su

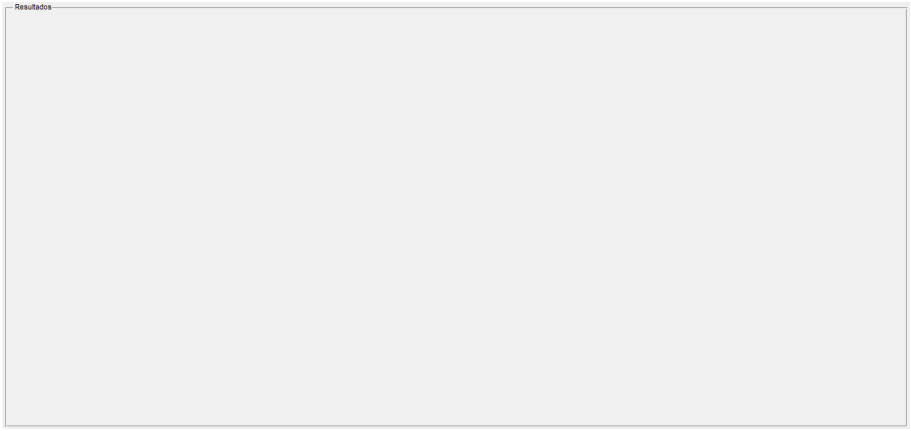
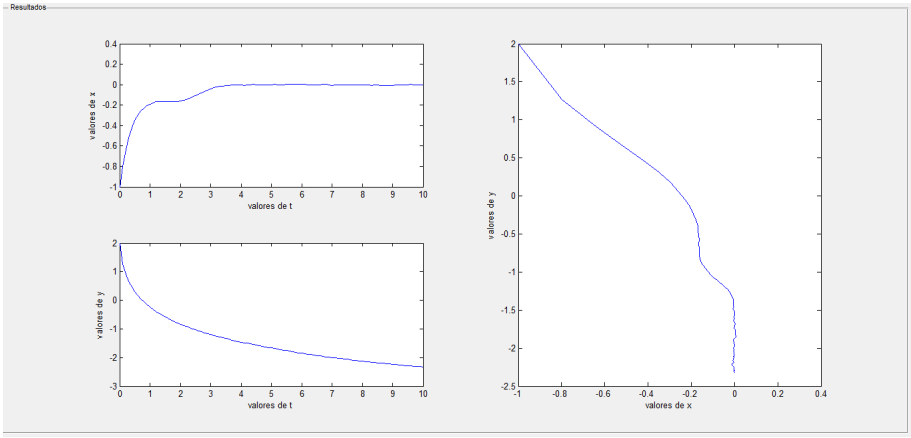


Figura 3.11: Panel de resultados

uso más común es representar los resultados obtenidos en ese momento y esto se hace automáticamente una vez que termina el cálculo. Las gráficas que se obtienen por defecto son la evolución de cada una de las componentes de la solución frente al tiempo y la evolución de cada una de las componentes de la solución frente a las otras, es decir, la trayectoria de la solución.

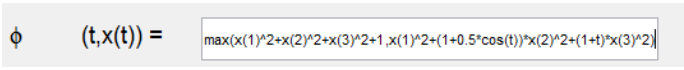
A continuación se presentan un par de ejemplos de problemas resueltos mediante la interfaz fráfica. Se muestran los potenciales introducidos en los cuadros de texto correspondientes y los resultados obtenidos en el panel de resultados.

Potencial introducido

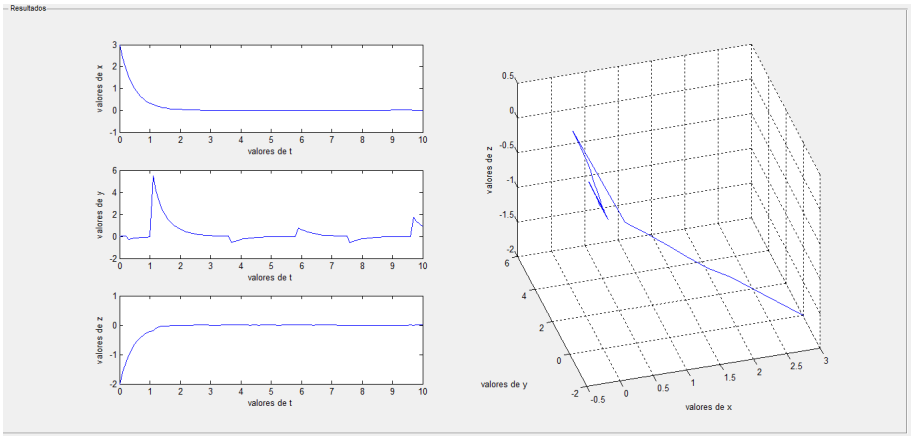


Resultados obtenidos

Figura 3.12: Ejemplo 1 con punto inicial (-1,2)



Potencial introducido



Resultados obtenidos

Figura 3.13: Ejemplo 2 con punto inicial (3,0,-2)

Capítulo 4

Problemas de persecución

Se estudiará en este capítulo un caso particular de problemas asociados a gradientes a los que hemos denominado *problemas de persecución*. Se expondrán en primer lugar los fundamentos teóricos para pasar a continuación a la implementación de su resolución mediante un código MATLAB.

4.1 Fundamentos teóricos

4.1.1 Caso estacionario

En el Ejemplo 2.7 del Capítulo 2 hemos visto cómo las trayectorias de las soluciones de $-\dot{\mathbf{x}}(t) \in \partial d_{\mathcal{B}}(\mathbf{x}(t))$, con $\mathcal{B}$ la bola unidad en $\mathbb{R}^2$, son rectas perpendiculares a la frontera de $\mathcal{B}$ que a partir del punto inicial alcanzan en tiempo finito el conjunto y se mantienen constantes.

Este resultado es cierto en general, ya que si $C \subset \mathbb{R}^N$ es un conjunto convexo y cerrado no vacío, se tiene que, para cada $\mathbf{x} \notin C$, la función distancia es diferenciable y

$$\nabla d_C(\mathbf{x}) = \frac{1}{d_C(\mathbf{x})} \left(\mathbf{x} - \mathrm{proj}_C(\mathbf{x})\right) \tag{4.1}$$

(ver [16, pág. 340]). Teniendo en cuenta, además, que

$$\mathrm{proj}_C \left(\mathbf{x} + \mu(\mathrm{proj}_C(\mathbf{x}) - \mathbf{x})\right) = \mathrm{proj}_C(\mathbf{x})$$

para cada $\mathbf{x} \notin C$, $0 < \mu < 1$, se tiene que la solución de

$$\left.\begin{array}{l} -\dot{\mathbf{x}}(t) \in \partial d_C(\mathbf{x}(t)) \\ \mathbf{x}(0) = \mathbf{x}^0 \end{array}\right\} \tag{4.2}$$

si $\mathbf{x}^0 \notin C$, es de la forma

$$\mathbf{x}(t) = \begin{cases} \mathbf{x}^0 - \dfrac{t}{d_C(\mathbf{x}^0)} \left(\mathbf{x}^0 - \mathrm{proj}_C(\mathbf{x}^0)\right), & 0 \leq t < d_C(\mathbf{x}^0) \\ \mathrm{proj}_C(\mathbf{x}^0), & t \geq d_C(\mathbf{x}^0) \end{cases} \tag{4.3}$$

Es decir, la solución de (4.2) proporciona la trayectoria óptima para alcanzar el conjunto C a partir de un punto arbitrario $\mathbf{x}^0$ en un tiempo finito e igual a la distancia del punto al conjunto.

4.1.2 Caso general

Cuando el conjunto C no está fijo, es decir, se tiene una aplicación que a cada t le asocia un conjunto convexo y cerrado no vacío, $C(t) \subset \mathbb{R}^N$, la situación no es tan simple. Pero, en cualquier caso, la ecuación (4.1) nos indica que la solución de

$$\left.\begin{array}{l} -\dot{\mathbf{x}}(t) \in \partial d_{C(t)}(\mathbf{x}(t)) \\ \mathbf{x}(0) = \mathbf{x}^0 \end{array}\right\} \tag{4.4}$$

si $\mathbf{x}(t) \notin C(t)$, verifica la igualdad

$$\dot{\mathbf{x}}(t) = \frac{1}{d_{C(t)}(\mathbf{x}(t))} \left(\mathrm{proj}_{C(t)}(\mathbf{x}(t)) - \mathbf{x}(t)\right)$$

es decir, la "velocidad" de la trayectoria determinada por $\mathbf{x}(\cdot)$ "apunta" en cada instante t hacia el conjunto $C(t)$. En otras palabras, la trayectoria de la solución de (4.4) va "persiguiendo" a los conjuntos $C(t)$. De aquí la denominación que hemos dado a este capítulo.

Aunque, como hemos dicho, las soluciones de (4.4) definen trayectorias que siguen a los conjuntos se plantean dos cuestiones que, en general, no tienen respuesta afirmativa:

- ¿Se alcanza a la familia de conjuntos en algún instante? Es decir, ¿existe $t^* > 0$ de forma que $\mathbf{x}(t^*) \in C(t^*)$?
- En ese caso, ¿la trayectoria de la solución se mantiene dentro de los conjuntos?, es decir, ¿para cada $t \geq t^*$ se tiene que $\mathbf{x}(t) \in C(t)$?

Cuando la primera cuestión tiene respuesta afirmativa se dice que el problema es *factible* (del inglés *feasible*). Si también la segunda es cierta el problema se denomina *sostenible* o *viable* (del inglés *sustainable* o *viable*).

El problema de identificar los sistemas factibles o sostenibles no es en absoluto sencilla. Veremos en el siguiente apartado una condición suficiente de sostenibilidad en función de la dinámica de los conjuntos $C(t)$.

4.1.3 Una condición suficiente de sostenibilidad

Dada una familia de conjuntos $C(t)$, se dice que es Lipschitz si existe una constante $\alpha > 0$ de forma que para cada s, t

$$\mathbf{dl}_{\mathcal{H}}(C(t), C(s)) \leq \alpha |t - s| \tag{4.5}$$

donde $\mathbf{dl}_{\mathcal{H}}$ es la distancia de Hausdorff entre conjuntos introducida en el Capítulo 1, página 7.

De [7, Th. 1] se deduce el siguiente resultado de sostenibilidad, que además permite modificar un sistema para hacerlo sostenible.

Teorema 4.1 *Sea $C(t)$ una familia de conjuntos cerrados y convexos no vacíos verificando la condición de Lipschitz en $[0,T]$.*

1. *Si $\alpha < 1$ y además $T(1-\alpha) > d_{C(0)}(\mathbf{x}^0)$, se tiene que (4.4) es sostenible. En particular, $\mathbf{x}(t) \in C(t)$ si $0 < t^* = d_{C(0)}(\mathbf{x}^0)/(1-\alpha) < t < T$.*

2. *En otro caso, sea $\alpha < \beta$, $T(\beta-\alpha) > d_{C(0)}(\mathbf{x}^0)$. Se tiene que el problema modificado*

$$\left.\begin{array}{l} -\dot{\mathbf{x}}(t) \in \beta \partial d_{C(t)}(\mathbf{x}(t)) \\ \mathbf{x}(0) = \mathbf{x}^0 \end{array}\right\} \tag{4.6}$$

   es sostenible.

Notar que, intuitivamente, lo que se ha hecho al modificar el problema para hacerlo sostenible es "aumentar" la velocidad de su trayectoria para que así pueda alcanzar a los conjuntos. En este segundo caso el tiempo a partir del cual la trayectoria del problema queda contenida en C es $t^* = d_{C(0)}(\mathbf{x}^0)/(\beta-\alpha)$.

4.1.4 Caso perturbado

En el caso del problema

$$\left.\begin{array}{l} -\dot{\mathbf{x}}(t) \in \partial d_{C(t)}(\mathbf{x}(t)) - \mathbf{F}(t,\mathbf{x}(t)) \\ \mathbf{x}(0) = \mathbf{x}^0 \end{array}\right\} \tag{4.7}$$

la sostenibilidad se ve dificultada por la existencia de un término fuente. Sin embargo es posible obtener un teorema similar al anterior teniendo el cuenta el efecto de $\mathbf{F}$.

Teorema 4.2 *Sea $C(t)$ una familia de conjuntos cerrados y convexos no vacíos verificando la condición de Lipschitz en $[0,T]$.*

1. *Si $\sup(|\mathbf{F}(t,\mathbf{x})| : \mathbf{x} \notin C(t)) < 1-\alpha$, para cada $0 \le t \le T$ y*

$$\int_0^T \delta(s)\, ds > d_{C(0)}(\mathbf{x}^0) \tag{4.8}$$

   con $0 < \delta(t) = 1 - \alpha - \sup(|\mathbf{F}(t,\mathbf{x})| : \mathbf{x} \notin C(t))$, se tiene que (4.7) es sostenible.

2. *En otro caso, si existen funciones β, δ positivas de forma que para cada $0 \le t \le T$, $\mathbf{x} \notin C(t)$,*

$$\beta(t,\mathbf{x}) \ge \alpha + \delta(t) + |\mathbf{F}(t,\mathbf{x})|$$

y se verifica (4.8), se tiene que el problema modificado

$$\left.\begin{array}{l} -\dot{\mathbf{x}}(t) \in \beta(t,\mathbf{x}(t))\partial d_{C(t)}(\mathbf{x}(t)) - \mathbf{F}(t,\mathbf{x}(t)) \\ \mathbf{x}(0) = \mathbf{x}^0 \end{array}\right\} \tag{4.9}$$

es sostenible.

3. *En ambos casos*

$$t^* = \min\left(0 < t < T : \int_0^t \delta(s)\,ds = d_{C(0)}(\mathbf{x}^0)\right)$$

4.2 Resolución numérica

Si tomamos como (P) el problema (4.4), su aproximado, considerando la regularización de Yosida de nivel $\lambda > 0$ de la subdiferencial de la función distancia será:

$$\left.\begin{array}{l} -\dot{\mathbf{x}}_\lambda(t) = \frac{1}{\lambda}\left(\mathbf{x}_\lambda(t) - J_\lambda(t,\mathbf{x}_\lambda(t))\right) \\ \mathbf{x}_\lambda(0) = \mathbf{x}^0 \end{array}\right\} \tag{$\mathcal{P}_\lambda$}$$

con $J_\lambda(\cdot)$ el campo resolvente, que en este caso toma la forma

$$J_\lambda(t,\mathbf{x}) = \underset{\mathbf{y}\in\mathbb{R}^N}{\operatorname{argmin}}\left(d_{C(t)}(\mathbf{y}) + \frac{1}{2\lambda}\left|\mathbf{y}-\mathbf{x}\right|^2\right) \tag{4.11}$$

Este problema presenta una dificultad técnica adicional a la hora de abordar su resolución numérica, dado que no conocemos de forma explícita, salvo en algunos casos especiales, el valor del potencial (la función distancia). Es por ello que no puede usarse la función `fminsearch` de MATLAB para evaluar la resolvente en cada paso del problema discretizado.

Una posible solución es tener en cuenta que para cada t y cada $\mathbf{x} \in \mathbb{R}^N$, la única solución del problema de minimización con restricciones

$$\underset{\mathbf{y}\in\mathbb{R}^N,\mathbf{z}\in C(t)}{\operatorname{argmin}}\left(|\mathbf{y}-\mathbf{z}| + \frac{1}{2\lambda}\left|\mathbf{y}-\mathbf{x}\right|^2\right) \tag{4.12}$$

es $(J_\lambda(t,\mathbf{x}), \operatorname{proj}_{C(t)}(J_\lambda(t,\mathbf{x})))$. Por tanto, para calcular la resolvente de la función distancia a un conjunto, basta con resolver el problema de minimización con restricciones (4.12) y tomar las N primeras componentes de la solución. Y aquí volvemos a encontrarnos con que la rutina `fminsearch` solamente resuelve problemas de optimización sin restricciones, por lo que tampoco puede calcular la resolvente a partir de (4.12).

Antes de ver con más detalle cómo se ha resuelto esta cuestión veamos los algoritmos modificados de los métodos de Euler y Runge-Kutta de orden cuatro para obtener soluciones aproximadas de ($\mathcal{P}_\lambda$).

4.2.1 Método de Euler

El algoritmo que sigue el método será básicamente igual al del caso general, solo que habrá que modificar el cálculo de la resolvente

---

Modificación Euler (problema de persecución)

2'. Definición del nuevo nodo:

$$\mathbf{y}_\lambda^{m,j} = \text{proj}\left(\underset{\mathbf{y}\in\mathbb{R}^N,\mathbf{z}\in C(t^{m,j})}{\text{argmin}}\left(|\mathbf{y}-\mathbf{z}| + \frac{1}{2\lambda}\left|\mathbf{y}-\mathbf{x}_\lambda^{m,j}\right|^2\right); \mathbb{R}^N\right)$$

donde proj$(;\mathbb{R}^N)$ selecciona las N primeras componentes

---

Al igual que en todos los casos anteriores, la resolución de $(\mathcal{P}_\lambda)$ mediante el método de Euler está programada en la función **euler.m**. El argumento de entrada que hay que incluir ahora para señalar que se trata de un problema de persecuciń es `dominiok`. Como se indicó cuando se explicaron todos los argumentos de entrada de la función, `dominiok` es un puntero que indica que se trata de un problema de persecución, por lo que se obvia la función `fun_prueba` y se considera como función potencial la distancia al conjunto definido (que obviamente puede variar con el tiempo).

Lo mayor diferencia es que ahora, en lugar de usar la función **moreau.m** para calcular la resolvente en cada iteración del bucle *for*, se ha creado una nueva función llamada **moreaudist.m**, que tiene la siguiente forma:

```
function [f,d]=moreaudist(t1,x)
```

Sus argumentos de entrada son:

- `t1`: escalar con el valor del instante de tiempo considerado.
- `x`: vector que contiene el valor de la solución en el instante de tiempo considerado.

y los de salida:

- `f`: vector con el valor de las diferentes componentes de la resolvente.
- `d`: escalar con el valor de la distancia de la solución al conjunto dado en el instante de tiempo considerado.

Además como variables de tipo `global` en el programa hay que introducir `dim`, que es un escalar con el número de dimensiones del problema considerado.

El objetivo fundamental de la función es el cálculo de la resolvente (variable `f`). Para su cálculo, y a fin de evitar los problemas mencionados por no conocer el

valor de la función distancia en cada punto, hemos usado la librería **solvopt.m**, un resolvedor de libre disposición (freeware) que permite calcular mínimos con o sin restricciones de funciones no necesariamente diferenciables (ver [12] para más detalles). La cabecera de la función es:

```
function [x,f,options]=solvopt(x,fun,grad,options,func,gradc)
```

y los argumentos de entrada que se usan son:

- `x`: vector con las coordenadas del punto de inicio de la iteración para la busqueda del mínimo. En este caso viene dado por la variable `x0`, cuyos primeros componentes son los del centro del dominio considerado (dados por la función `centro` en la que está programada la expresión del centro del dominio a lo largo del tiempo) y los últimos son el valor de la solución en el punto anterior.
- `fun`: nombre de la función donde se tendrá definida la función objetivo. En este caso esta función es `funcionobjetivo` y en ella está programada la expresión de la resolvente.
- `options`: vector con los valores que se quieren usar para los parámetros internos del programa como el número máximo de iteraciones, la tolerancia de la solución y demás. Se pueden ver todos en [12]. Sólo mencionar que en este caso se ha usado `[-1 1e-4 1e-6 15000 -1]` y el último valor (`-1`) se utiliza para que el programa no muestre mensajes con los resultados intermedios de la iteración para no llenar el espacio de trabajo de mensajes innecesarios.
- `func`: nombre de la función donde se tendrán definidas las restricciones del problema. Estas restricciones son precisamente la expresión del conjunto que se quiera considerar y por lo tanto se deberán cambiar para cada problema que se quiera estudiar. Esto está programado en la función `funcionrestricciones` y ésta será la que habrá que modificar para cada problema estudiado.

Los argumentos de salida utilizados son:

- `x`: vector con las coordenadas del lugar donde se produce el mínimo de la función objetivo. En este caso este vector tiene más componentes de las que nos interesan y sólo tomaremos las `dim` últimas componentes, que son las que realmente se corresponden con la solución.

La función **moreaudist.m** aparte de obtener el valor de la resolvente, también proporciona el valor de la distancia de la solución al conjunto dado en el instante de tiempo considerado (variable `d`). Para esto se vuelve a utilizar la función **solvopt.m**. Ahora se usa como `funcionobjetivo` la expresión de $d_{\mathcal{C}(t)}(\mathbf{x}(t))$ (programada en la función `funciondistancia`) y como `funcionrestricciones` la misma de antes. Señalar que ahora no se toma el primer argumento de salida de la función **solvopt.m** (lugar donde se produce el mínimo, `x`) como antes sino el segundo argumento de salida (valor de ese mínimo, `f`).

4.2.2 Método de RK4

La diferencia en los algoritmos respecto al caso general está ahora simplemente en el cálculo de la resolvente que debe hacerse en cuatro ocasiones en cada iteración para obtener los correspondientes coeficientes.

Modificación RK4 (problema de persecución)

2'. Cálculo de los coeficientes:

$$\mathbf{k}_{\lambda,1}^{m,j} = \frac{1}{\lambda}\left(\mathbf{x}_\lambda^{m,j} - J_\lambda(t^{m,j}, \mathbf{x}_\lambda^{m,j})\right)$$

$$\mathbf{k}_{\lambda,2}^{m,j} = \frac{1}{\lambda}\left(\mathbf{x}_\lambda^{m,j} - \frac{h_m}{2}\mathbf{k}_{\lambda,1}^{m,j} - J_\lambda(t^{m,j} + \frac{h_m}{2}, \mathbf{x}_\lambda^{m,j} - \frac{h_m}{2}\mathbf{k}_{\lambda,1}^{m,j})\right)$$

$$\mathbf{k}_{\lambda,3}^{m,j} = \frac{1}{\lambda}\left(\mathbf{x}_\lambda^{m,j} - \frac{h_m}{2}\mathbf{k}_{\lambda,2}^{m,j} - J_\lambda(t^{m,j} + \frac{h_m}{2}, \mathbf{x}_\lambda^{m,j} - \frac{h_m}{2}\mathbf{k}_{\lambda,2}^{m,j})\right)$$

$$\mathbf{k}_{\lambda,4}^{m,j} = \frac{1}{\lambda}\left(\mathbf{x}_\lambda^{m,j} - h_m\mathbf{k}_{\lambda,3}^{m,j} - J_\lambda(t^{m,j} + h_m, \mathbf{x}_\lambda^{m,j} - h_m\mathbf{k}_{\lambda,3}^{m,j})\right)$$

donde

$$J_\lambda(t, \mathbf{x}) = \mathrm{proj}\left(\underset{\mathbf{y}\in\mathbb{R}^N, \mathbf{z}\in C(t)}{\operatorname{argmin}}\left(|\mathbf{y}-\mathbf{z}| + \frac{1}{2\lambda}|\mathbf{y}-\mathbf{x}|^2\right); \mathbb{R}^N\right)$$

Se utiliza para solucionar este tipo de problema la función **rk4.m** y, al igual que en el método de Euler, habrá que incluir como argumento de entrada `dominiok` para señalar que se trata de un problema de persecución. Dentro del bucle *for* solo variará, al igual que en **euler.m**, la resolución de la resolvente que se hace también con la función **moreaudist.m**. Al ser todo su funcionamiento análogo al que se ha expuesto en el método de Euler no se repetirá la explicación.

4.2.3 Otros casos

Al igual que en el caso general, se pueden considerar diversas clases de problemas donde aparecen subdiferenciales de funciones distancia a conjuntos. En particular problemas perturbados por un término fuente como (4.7) o (4.9).

También puede añadirse un segundo potencial y considerar el problema bipotencial

$$-\dot{\mathbf{x}}(t) \in \beta(t, \mathbf{x}(t))\partial d_{C(t)}(\mathbf{x}(t)) + \partial\varphi(t, \mathbf{x}(t)) \tag{4.13}$$

al que puede añadirse un término fuente

$$-\dot{\mathbf{x}}(t) \in \beta(t, \mathbf{x}(t))\partial d_{C(t)}(\mathbf{x}(t)) + \partial\varphi(t, \mathbf{x}(t))) - \mathbf{F}(t, \mathbf{x}(t)) \tag{4.14}$$

Las modificaciones necesarias en los algoritmos para obtener soluciones de estos tipos de problemas son análogas a las realizadas en el caso general, por lo que no se volverán a comentar. No obstante hemos incluidos los códigos correspondientes en el Anexo de la memoria.

4.2.4 Interfaz gráfica

Al igual que se ha hecho con el caso general, se ha desarrollado en la interfaz gráfica un panel de resultados específico para los problemas de persecución. Su aspecto general se presenta en la Figura 4.1 y comparte muchas características con el panel de introducción de datos para el caso general.

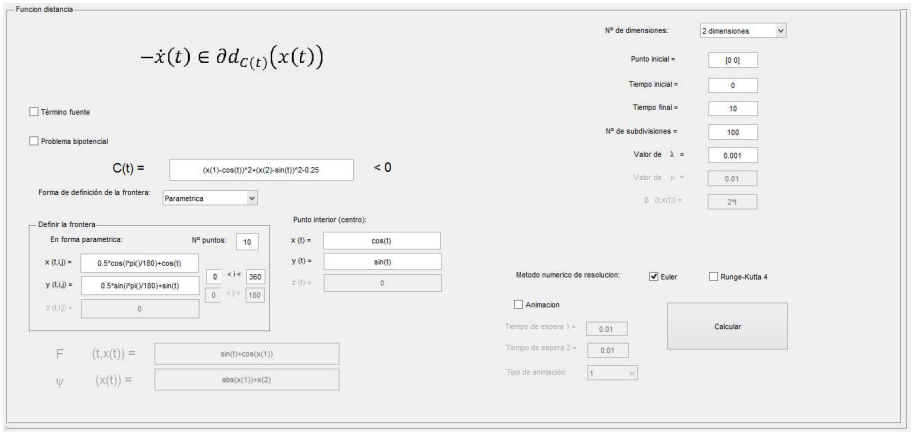


Figura 4.1: Panel de introducción de datos en problemas de persecución

En este caso, las diferentes expresiones matemáticas del problema también varían dependiendo del caso elegido mediante los botones de selección, pero son ligeramente diferentes a las del caso general. Ahora son las dadas en la Figura 4.2.

$-\dot{x}(t) \in \partial d_{C(t)}(x(t))$

Caso general

$-\dot{x}(t) \in \beta(t,x(t))\partial d_{C(t)}(x(t)) - F(t,x(t))$

Caso con término fuente

$-\dot{x}(t) \in \beta(t,x(t))\partial d_{C(t)}(x(t)) + \partial\Psi(t,x(t))$

Caso bipotencial

$-\dot{x}(t) \in \beta(t,x(t))\partial d_{C(t)}(x(t)) + \partial\Psi(t,x(t)) - F(t,x(t))$

Caso bipotencial con término fuente

Figura 4.2: Formulación matemática del problema de persecución

El parámetro quizá más importante y que diferencia este tipo de problemas del caso general es el dominio $C(t)$. Éste se define en un cuadro de texto en la interfaz gráfica al igual que el resto de funciones potenciales (Figura 4.3).

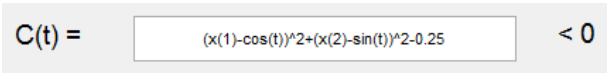


Figura 4.3: Ejemplo de definición de dominio

Un elemento nuevo que aparece en este tipo de problemas es la definición de un punto que se encuentre siempre en el interior del dominio para que comience en él la iteración para la búsqueda del mínimo que hay que hacer. Arbitrariamente se ha elegido para este punto el centro del dominio, aunque valdría cualquiera. Cabe destacar que esto no es indispensable, puesto que aunque se fijara un punto fijo como punto de comienzo de la iteración, debido a la convexidad del problema, el resolvedor **solvopt.m** seguiría funcionando correctamente. No obstante, al usar un punto que se encuentre siempre dentro del dominio (su centro por ejemplo) se agiliza bastante el proceso de cálculo. En la interfaz gráfica, este punto se introduce mediante los cuadros de texto mostrados en la Figura 4.4.

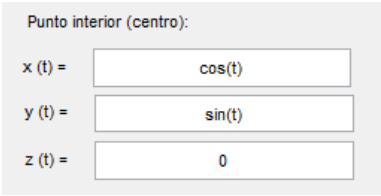


Figura 4.4: Coordenadas del centro del dominio

El apartado de elección del método numérico a utilizar y el botón para comenzar el cálculo es similar al del caso general, encontrándose la mayor diferencia cuando se elige realizar una animación de los resultados. En este caso, si se marca la opción de animación, aparecerán dos tiempos de espera distintos para definir y aparecen 3 tipos distintos de animación para elegir (Figura 4.5).

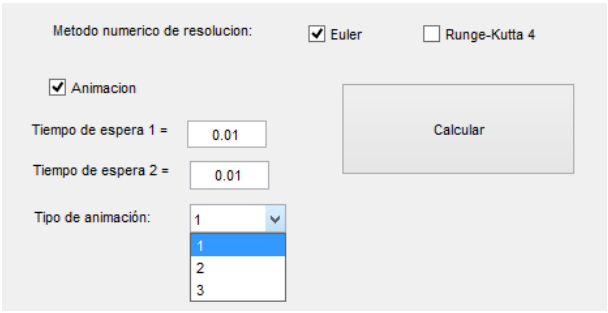


Figura 4.5: Opciones de resolución

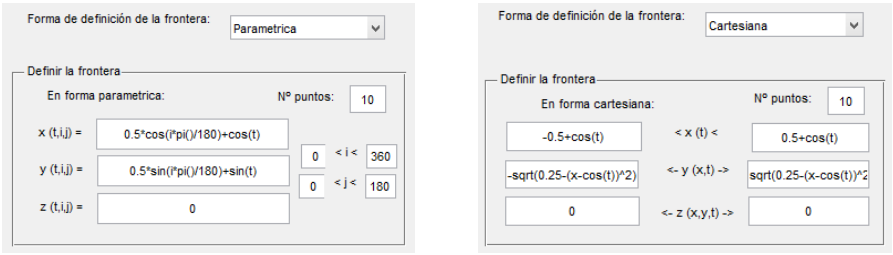
Estos dos tiempos distintos se corresponden, el primero de ellos al tiempo

de espera para la representación de cada una de las componentes de la solución frente al tiempo, cada una de las componentes de la solución frente a las otras y la distancia de la solución al conjunto frente al tiempo. El segundo de estos tiempos se corresponde al tiempo de espera para la representación conjunta de la evolución del dominio y la solución frente al tiempo.

Es precisamente en esta representación conjunta del dominio y la solución donde se ha prestado una mayor atención en programación y se han desarrollado diversas opciones para permitir poder elegir la visualización más adecuada en cada caso. Las 3 opciones de animación son:

- **Tipo 1**: Muestra tanto en 2D como en 3D en cada instante la representación del dominio y la solución a la vez. Cuando pasa al siguiente instante de tiempo se borran los anteriores y se representan los nuevos, por lo que sólo se muestra lo que sucede en cada instante.
- **Tipo 2**: En 2D muestra una gráfica con 3 ejes que representan las dos componentes de la solución y el tiempo, donde se van representando en cada punto del eje temporal el dominio y la solución. Así pues, se tiene finalmente la trayectoria seguida por la solución y por el dominio. En 3D, al no ser posible representar 3 componentes espaciales y una temporal en una misma gráfica, se representan sólo las 3 variables espaciales para cada instante del tiempo, por lo que finalmente se tiene la trayectoria seguida por la solución.
- **Tipo 3**: Muestra tanto en 2D como en 3D en cada instante la representación del dominio y la solución a la vez. Cuando pasa al siguiente instante de tiempo se borra la representación anterior del dominio pero se mantiene la de la solución, por lo que sólo se muestra el dominio en cada instante pero se va almacenando la trayectoria seguida por la solución.

Para poder dibujar el dominio hay que introducir al programa la ecuación de la frontera del mismo. Esto se hace en los cuadros de texto mostrados en la Figura 4.6. Esta ecuación de la frontera se puede introducir en coordenadas



Coordenadas paramétricas Coordenadas cartesianas

Figura 4.6: Formas de definición de la frontera

paramétricas o en coordenadas cartesianas. Esto se elige en un menú desplegable. Además se elige el número de puntos con los que se quiere dibujar este

dominio. A mayor número de puntos, tarda más el cálculo pero se obtiene un resultado más detallado.

En el caso de los problemas de persecución, y si se ha marcado la opción de animación, una vez que se pulsa el botón *Calcular*, vuelve a aparecer la barra de progreso de cálculo que aparecía en el caso general. Ahora tras ésta aparece una barra de progreso de dibujo (Figura 4.7) mientras se calculan los puntos del dominio necesarios para la representación.

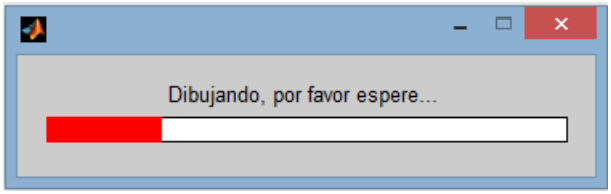


Figura 4.7: Progreso del dibujo

Señalar por último que ahora una vez que el programa termina de calcular, al igual que antes, pasará automáticamente al panel de resultados y comenzará su representación gráfica. Las gráficas que se generarán serán las mismas con la adición de una gráfica que represente la evolución de la distancia de la solución al dominio considerado a lo largo del tiempo. En el caso de que esté marcada la opción de animación, se presentará además una animación donde aparecerá la evolución de la solución y el dominio a lo largo del tiempo como se puede ver en la Figura 4.8.

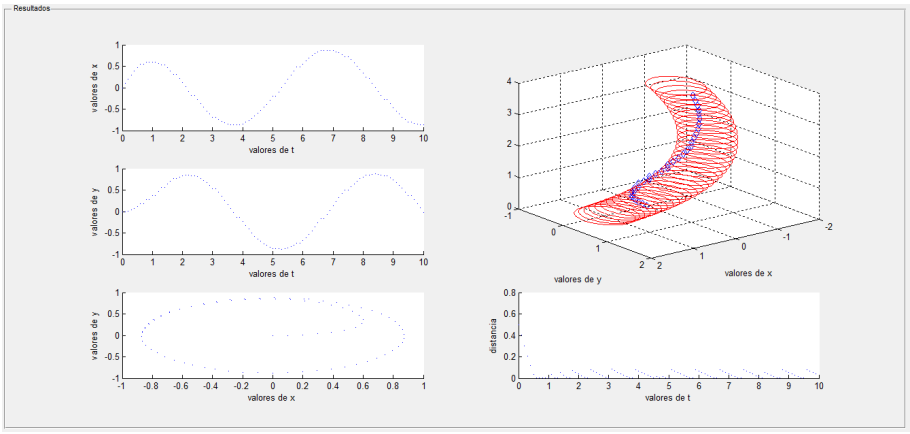


Figura 4.8: Ejemplo de gráficas de solución con animación

4.2.5 Simulaciones

Presentamos ahora una serie de ejemplos prácticos en los que se ha usado el código desarrollado en el proyecto para resolver diferentes problemas de persecución.

Ejemplo 4.1 Dada la familia de círculos con centro y radio variables

$$C(t) = \left\{(x,y) \in \mathbb{R}^2 : (x-\cos(t))^2 + (y-\text{sen}(t))^2 \leq 0.5 + 0.25\cos(2t)\right\}$$

se considera el problema (4.4) asociado. Para resolverlo usamos el método RK4

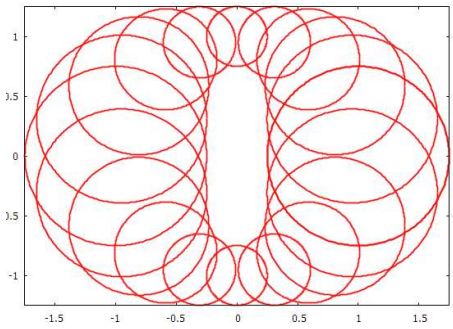
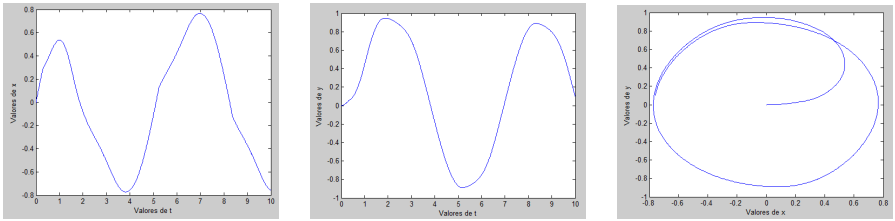


Figura 4.9: Círculos con centro y radio móviles

de acuerdo con los parámetros de la siguiente tabla

Parámetro	Valor	Parámetro	Valor
a	0	x0	0
b	10	y0	0
m	1000	x_centro	cos(t)
lambda	0.001	y_centro	sin(t)

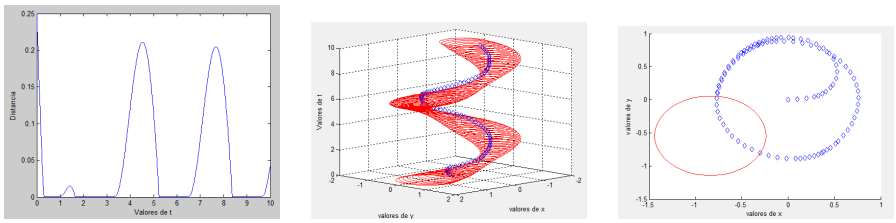
Se muestran a continuación las gráficas de las componentes y de la trayectoria



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 4.10: Ejemplo 4.1 con punto inicial (0,0)

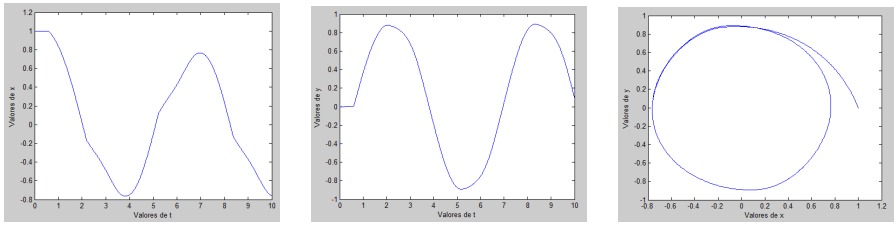
También se muestran la gráfica de la distancia de la solución a los conjuntos en cada instante de tiempo y la evolución temporal conjunta del dominio y la solución



Distancia $d_{C(t)}(\mathbf{x}(t))$ Conjunto y solución

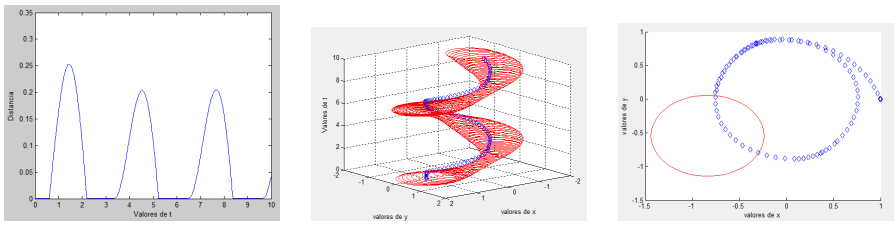
Figura 4.11: Ejemplo 4.1 con punto inicial (0,0)

Repetimos la simulación manteniendo todos los parámetros y modificando únicamente el punto inicial, que en este caso es $(1, 0)$. Se muestran las mismas salidas que en el caso anterior.



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 4.12: Ejemplo 4.1 con punto inicial (1,0)



Distancia $d_{C(t)}(\mathbf{x}(t))$ Conjunto y solución

Figura 4.13: Ejemplo 4.1 con punto inicial (1,0)

Ejemplo 4.2 Consideremos ahora un caso en que la frontera de los conjuntos no es "suave", por ejemplo

$$C(t) = \{(x, y) \in \mathbb{R}^2 : |2\cos(t) - x| + |2\,\mathrm{sen}(t) - y| \leq 1\}$$

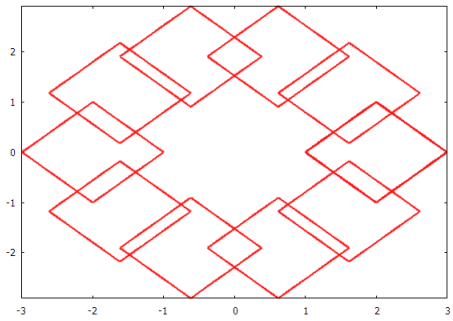


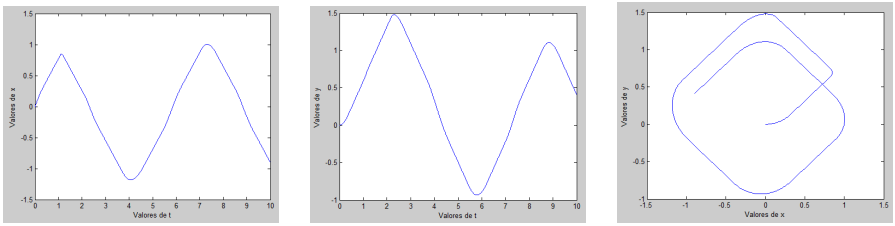
Figura 4.14: Cuadrados con centro móvil

Usamos el algoritmo de Moreau-Yosida junto con el método RK4, con los parámetros

Parámetro	Valor	Parámetro	Valor
a	0	x_centro	2*cos(t)
b	10	y_centro	2*sin(t)
m	1000		
lambda	0.001		

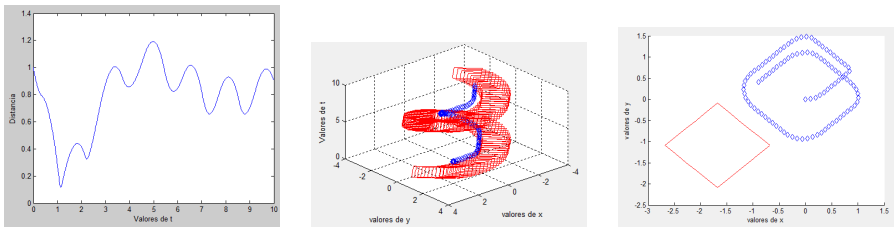
para resolver el problema (4.4) para la familia de cuadrados, obteniendo las siguientes gráficas para los puntos iniciales:

- $(x_0, y_0) = (0, 0)$



$x(t)$ $y(t)$ $(x(t), y(t))$

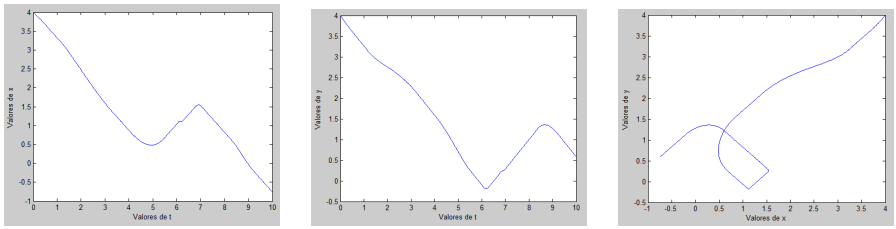
Figura 4.15: Ejemplo 4.2 con punto inicial (0,0)



Distancia $d_{C(t)}(\mathbf{x}(t))$ Conjunto y solución

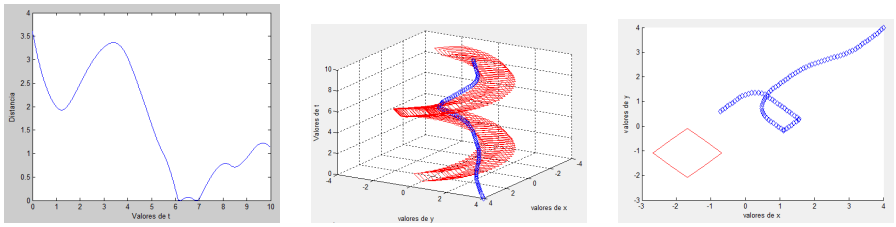
Figura 4.16: Ejemplo 4.2 con punto inicial (0,0)

- $(x_0, y_0) = (4, 4)$



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 4.17: Ejemplo 4.2 con punto inicial (4,4)



Distancia $d_{C(t)}(\mathbf{x}(t))$ Conjunto y solución

Figura 4.18: Ejemplo 4.2 con punto inicial (4,4)

Ejemplo 4.3 Consideremos ahora el mismo problema, donde ahora la familia de conjuntos son círculos que se alejan hacia la derecha,

$$C(t) = \left\{(x, y) \in \mathbb{R}^2 : (x - t)^2 + (y - \text{sen}(t))^2 \leq 1\right\}$$

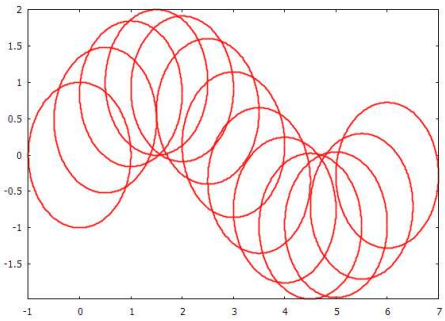


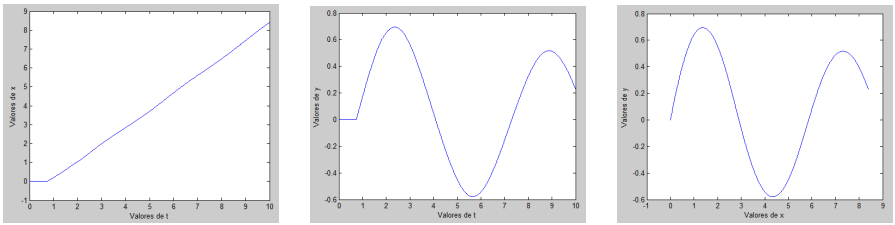
Figura 4.19: Conjuntos alejándose hacia la derecha

Usamos los parámetros

Parámetro	Valor	Parámetro	Valor
a	0	x_centro	t
b	10	y_centro	sin(t)
M	1000		
lambda	0.001		

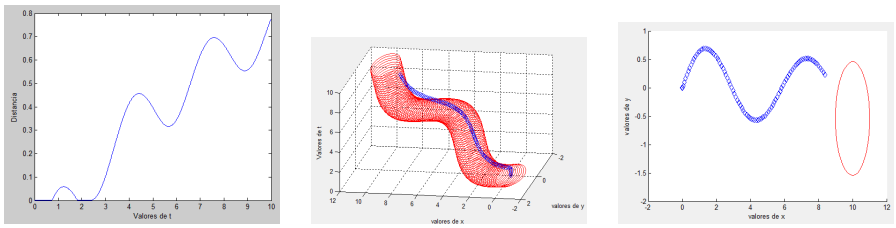
y resolvemos el problema para tres condiciones iniciales distintas:

- $(x_0, y_0) = (0, 0)$



$x(t)$ $y(t)$ $(x(t), y(t))$

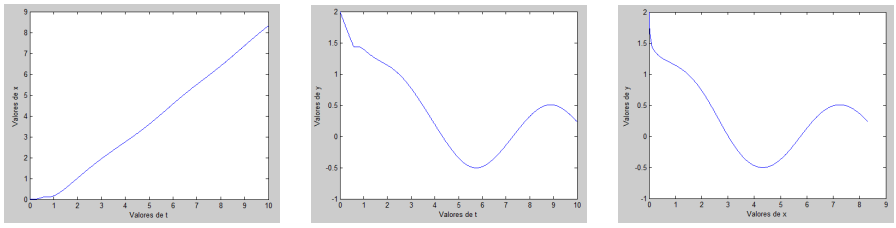
Figura 4.20: Ejemplo 4.3 con punto inicial (0,0)



Distancia $d_{C(t)}(\mathbf{x}(t))$ Conjunto y solución

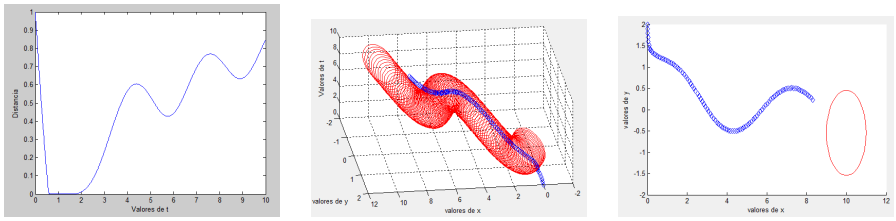
Figura 4.21: Ejemplo 4.3 con punto inicial (0,0)

- $(x_0, y_0) = (0, 2)$



$x(t)$ $y(t)$ $(x(t), y(t))$

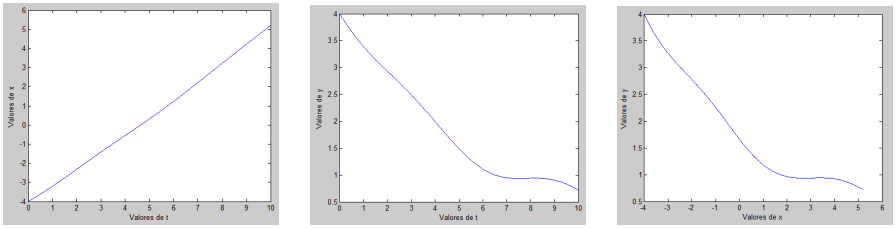
Figura 4.22: Ejemplo 4.3 con punto inicial (0,2)



Distancia $d_{C(t)}(\mathbf{x}(t))$ Conjunto y solución

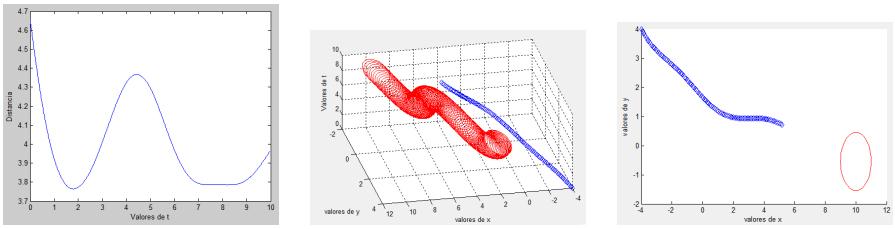
Figura 4.23: Ejemplo 4.3 con punto inicial (0,2)

- $(x_0, y_0) = (-4, 4)$



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 4.24: Ejemplo 4.3 con punto inicial (-4,4)



Distancia $d_{C(t)}(\mathbf{x}(t))$ Conjunto y solución

Figura 4.25: Ejemplo 4.3 con punto inicial (-4,4)

Ejemplo 4.4 Finalmente consideramos la familia de elipses

$$C(t) = \left\{ (x, y) \in \mathbb{R}^2 : \frac{x^2}{(2 + \cos(t))^2} + \frac{y^2}{(1 + \text{sen}^2(t))^2} \leq 1 \right\}$$

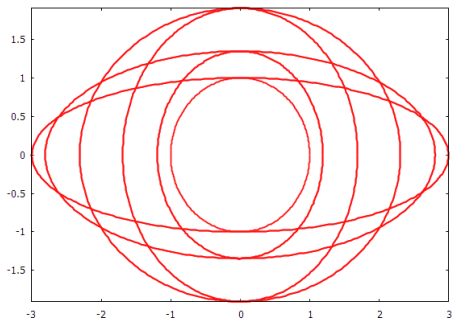


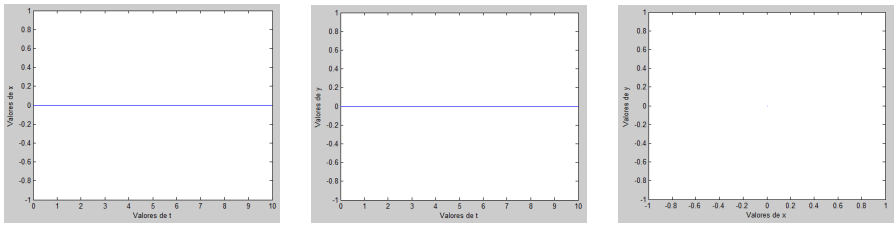
Figura 4.26: Elipses de semiejes variables

Usamos los parámetros

Parámetro	Valor	Parámetro	Valor
a	0	x_centro	0
b	10	y_centro	0
m	1000		
lambda	0.001		

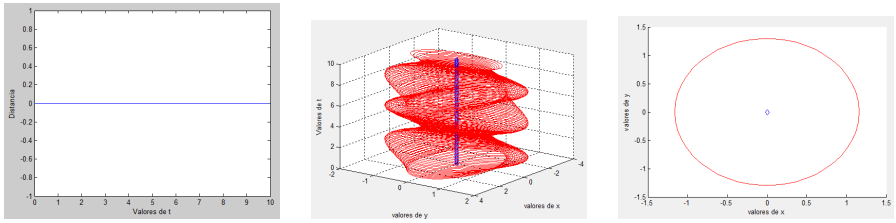
para obtener simulaciones de la inclusión asociada a la distancia a $C(t)$ para diferentes puntos iniciales.

- $(x_0, y_0) = (0, 0)$



$x(t)$ $y(t)$ $(x(t), y(t))$

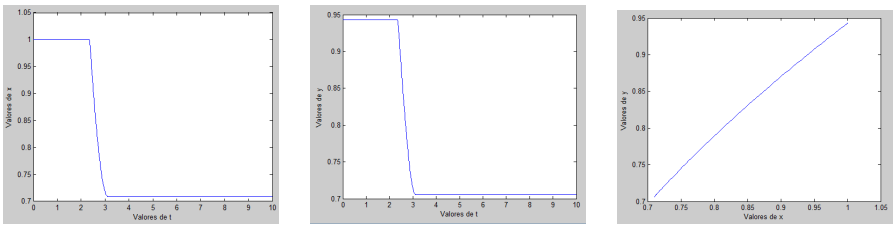
Figura 4.27: Ejemplo 4.4 con punto inicial (0,0)



Distancia $d_{C(t)}(\mathbf{x}(t))$ Conjunto y solución

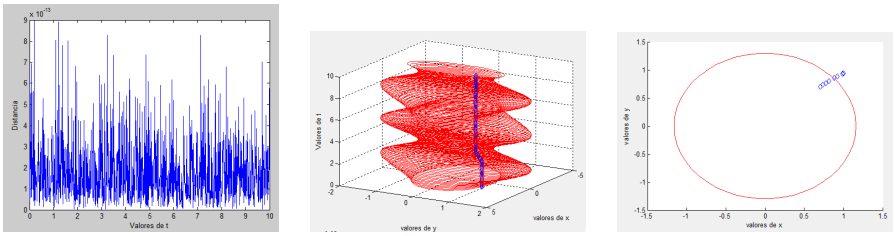
Figura 4.28: Ejemplo 4.4 con punto inicial (0,0)

- $(x_0, y_0) = (1, 2\sqrt{2}/3)$



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 4.29: Ejemplo 4.4 con punto inicial $(1, 2\sqrt{2}/3)$



Distancia $d_{C(t)}(\mathbf{x}(t))$ Conjunto y solución

Figura 4.30: Ejemplo 4.4 con punto inicial $(1, 2\sqrt{2}/3)$

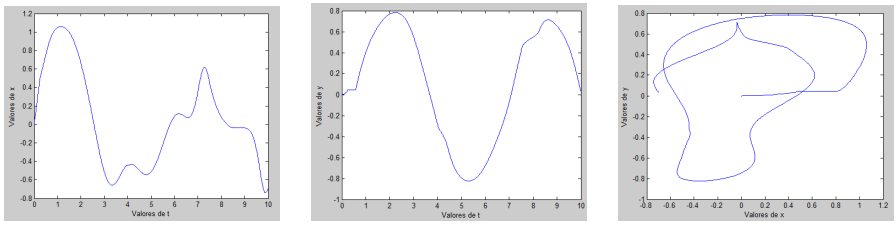
Ejemplo 4.5 Consideremos un caso de la forma (4.7), con

$$C(t) = \left\{(x, y) \in \mathbb{R}^2 : (x - \cos(t))^2 + (y - \operatorname{sen}(t))^2 \leq 0.5\right\}$$

y término fuente $\mathbf{F}(t, x, y) = (\cos(ty), 0)$. Utilizamos los siguientes parámetros

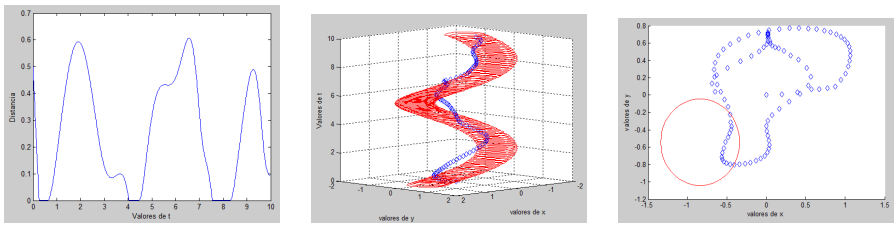
Parámetro	Valor	Parámetro	Valor
a	0	x0	0
b	10	y0	0
m	1000	x_centro	cos(t)
lambda	0.001	y_centro	sin(t)

para obtener las soluciones



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 4.31: Ejemplo 4.5 con punto inicial (0,0)



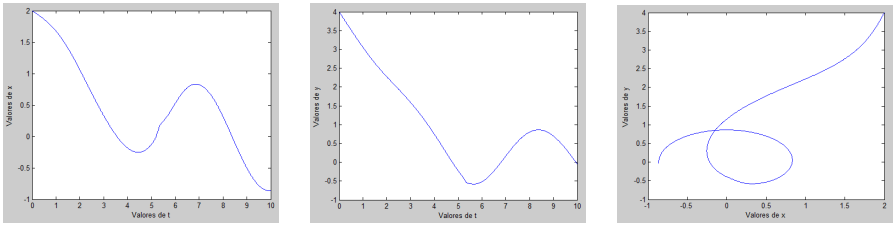
Distancia $d_{C(t)}(\mathbf{x}(t))$ Conjunto y solución

Figura 4.32: Ejemplo 4.5 con punto inicial (0,0)

Modificamos ahora el término fuente tomando

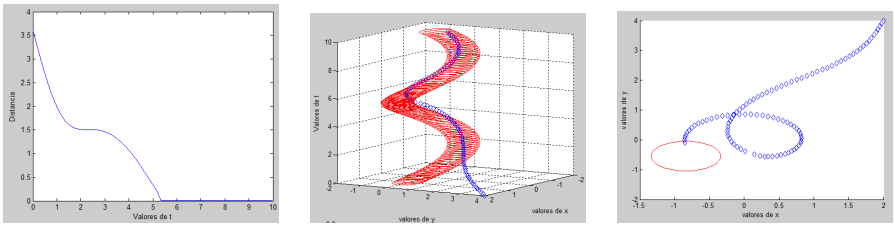
$$\mathbf{G}(t,x,y) = \begin{cases} (0,0), & \text{si } (x,y) \in C(t) \text{ o } d_{C(t)}(x,y) \geq 0.1 \\ (-x+\cos(t), -y+\text{sen}(t))\,, & \text{en otro caso} \end{cases}$$

Cambiamos también las condiciones iniciales, empezando en $(2,4)$ y mantenemos el resto de los parámetros en la simulación, obteniendo



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 4.33: Ejemplo 4.5 con punto inicial (2,4)



Distancia $d_{C(t)}(\mathbf{x}(t))$ Conjunto y solución

Figura 4.34: Ejemplo 4.5 con punto inicial (2,4)

Capítulo 5

Conclusiones y expectativas

En este trabajo se ha desarrollado un código MATLAB que permite implementar computacionalmente el algoritmo de Moreau-Yosida para la resolución aproximada de inclusiones subdiferenciales. Dicho código ha sido validado de forma satisfactoria mediante múltiples ejemplos en los que ha funcionado de forma eficiente proporcionando resultados que concuerdan con los desarrollos teóricos y con otros resultados conultados en la bibliografía. Además dicho código:

- Es altamente flexible ya que nos ha permitido resolver diferentes tipos de problemas (potenciales variables, bipotenciales, problemas perturbados,...)
- Es independiente de las dimensiones del problema, al estar programado de forma vectorial
- Está integrado en una completa interfaz gráfica que permite su manejo de forma sencilla y cómoda
- Proporciona una rica y variada información sobre el problema (distintos tipos de gráficas, ficheros exportables de datos

La labor de programación y desarrollo del entorno gráfico ha sido especialmente compleja y laboriosa, sin embargo ha permitido integrar con éxito todos los códigos desarrollados.

Como posibles ampliaciones y desarrollos del presente trabajo en el futuro cabe mencionar:

- La integración de métodos numéricos para ecuaciones diferenciales ordinarias distintos de los de Euler y Runge-Kutta en el algoritmo de Moreau-Yosida permitiría obtener mejores resultados en algunos problemas concretos.
- Usar el código para estudiar de forma detallada problemas concretos de ingeniería. En particular podría ser interesante su uso para simular numé-

ricamente el comportamiento de materiales compuestos frente a la acción de fuerzas.

- A pesar de que estos métodos están limitados por la convexidad del portencial, pueden aportar información en otros casos, por ejemplo estudiando problemas covexificados o relajados asociados.

- En esta memoria nos hemos limitado al caso de dimensión finita, pero la regularización de Yosida ha sido usada de forma exitosa para estudiar teóricamente problemas de dimensión infinita, en particular ecuaciones en derivadas parciales donde aparecen términos con discontinuidades. Esto abre la expectativa de usar estas ideas para abordar el estudio numérico de algunos problemas de este tipo.

Capítulo 6

Anexos

6.1 Algoritmos de resolución

6.1.1 euler.m

```
function [x,d]=euler(fun_prueba,x0,a,b,m,lambda,term_fuente,fun_prueba_2,
nu,beta,dominiok)

global dim T lambda1

h=(b-a)/m;
T=a:h:b;
x=zeros(m+1,dim);
d=zeros(m,1);
x(1,:)=x0(1,:);
lambda1=lambda;


u=waitbar(0,'Calculando, por favor espere...');
if strcmp(class(dominiok),'function_handle')&&ischar(term_fuente)&&...
        ischar(fun_prueba_2)
    for i=1:m
        t=T(i);
        [f,dist]=moreaudist(t,x(i,:));
        x(i+1,:)=x(i,:)-h/lambda*(x(i,:)-f);
        d(i)=dist;
        waitbar(i/m);
    end

elseif strcmp(class(dominiok),'function_handle')&&strcmp(class(term_fuente),
'function_handle')&&...
        strcmp(class(fun_prueba_2),'function_handle')
```

```
    for i=1:m
        t=T(i);
        [f,dist]=moreaudist(t,x(i,:));
        x(i+1,:)=x(i,:)-h/lambda*beta(t,x(i,:))*(x(i,:)-f)+h*term_fuente(t,
x(i,:))...
            -h/nu*(x(i,:)-moreau(t,x(i,:),x0,nu,fun_prueba_2));
        d(i)=dist;
        waitbar(i/m);
    end

elseif strcmp(class(dominiok),'function_handle')&&ischar(term_fuente)&&...
        strcmp(class(fun_prueba_2),'function_handle')
    for i=1:m
        t=T(i);
        [f,dist]=moreaudist(t,x(i,:));
        x(i+1,:)=x(i,:)-h/lambda*beta(t,x(i,:))*(x(i,:)-f)...
            -h/nu*(x(i,:)-moreau(t,x(i,:),x0,nu,fun_prueba_2));
        d(i)=dist;
        waitbar(i/m);
    end

elseif strcmp(class(dominiok),'function_handle')&& strcmp(class(term_fuente),
'function_handle')&&...
        ischar(fun_prueba_2)
    for i=1:m
        t=T(i);
        [f,dist]=moreaudist(t,x(i,:));
        x(i+1,:)=x(i,:)-h/lambda*beta(t,x(i,:))*(x(i,:)-f)+h*term_fuente(t,
x(i,:));
        d(i)=dist;
        waitbar(i/m);
    end

elseif strcmp(class(term_fuente),'function_handle')&&...
        strcmp(class(fun_prueba_2),'function_handle')
    for i=1:m
        t=T(i);
        x(i+1,:)=x(i,:)-h/lambda*(x(i,:)-moreau(t,x(i,:),x0,lambda,
 fun_prueba))-h/nu*beta(t,x(i,:))*(x(i,:)-moreau(t,x(i,:),
 x0,nu,fun_prueba_2))+h*term_fuente(t,x(i,:));
        waitbar(i/m);
    end

elseif strcmp(class(fun_prueba_2),'function_handle')&&...
       ischar(term_fuente)
    for i=1:m
```

```
        t=T(i);
        x(i+1,:)=x(i,:)-h/lambda*(x(i,:)-moreau(t,x(i,:),x0,lambda,
fun_prueba))-h/nu*beta(t,x(i,:))*(x(i,:)-moreau(t,x(i,:),
x0,nu,fun_prueba_2));
        waitbar(i/m);
    end

elseif strcmp(class(term_fuente),'function_handle')&&...
        ischar(fun_prueba_2)
    for i=1:m
        t=T(i);
        x(i+1,:)=x(i,:)-h/lambda*(x(i,:)-moreau(t,x(i,:),x0,lambda,
fun_prueba))+h*term_fuente(t,x(i,:));
        waitbar(i/m);
    end

elseif ischar(term_fuente)&& ischar(fun_prueba_2)
    for i=1:m
        t=T(i);
        x(i+1,:)=x(i,:)-h/lambda*(x(i,:)-moreau(t,x(i,:),x0,lambda,
fun_prueba));
        waitbar(i/m);
    end
end
close (u)
```

6.1.2 rk4.m

```
function [x,d]=rk4(fun_prueba,x0,a,b,m,lambda,term_fuente,fun_prueba_2,
nu,beta,dominiok)

global dim T lambda1

h=(b-a)/m;
T=a:h:b;
x=zeros(m+1,dim);
d=zeros(m,1);
x(1,:)=x0(1,:);
lambda1=lambda;


u=waitbar(0,'Calculando, por favor espere...');
if strcmp(class(dominiok),'function_handle')&&ischar(term_fuente)&&...
        ischar(fun_prueba_2)
    for i=1:m
        t=T(i);
```

```
        [f,dist]=moreaudist(t,x(i,:));
        d(i)=dist;
        k1 = -1/lambda*(x(i,:)-f);
        k2 = -1/lambda*(x(i,:)+h/2*k1-moreaudist(t+h/2,x(i,:)+h/2*k1));
        k3 = -1/lambda*(x(i,:)+h/2*k2-moreaudist(t+h/2,x(i,:)+h/2*k2));
        k4 = -1/lambda*(x(i,:)+h*k3-moreaudist(t+h,x(i,:)+h*k3));
        x(i+1,:)=x(i,:)+h/6*(k1+2*(k2+k3)+k4);
        waitbar(i/m);
    end

elseif strcmp(class(dominiok),'function_handle')&&strcmp(class
(term_fuente),'function_handle')&&strcmp(class(fun_prueba_2),
'function_handle')
    for i=1:m
        t=T(i);
        [f,dist]=moreaudist(t,x(i,:));
        d(i)=dist;
        k1 = -beta(t,x(i,:))/lambda*(x(i,:)-f)...
            -1/nu*(x(i,:)-moreau(t,x(i,:),x0,nu,fun_prueba_2))...
            +term_fuente(t,x(i,:));
        k2 = -beta(t+h/2,x(i,:)+h/2)/lambda*(x(i,:)+h/2*k1-moreaudist
(t+h/2,x(i,:)+h/2*k1))-1/nu*(x(i,:)+h/2*k1...
            -moreau(t,x(i,:)+h/2*k1,x0,nu,fun_prueba_2))...
            +term_fuente(t+h/2,x(i,:)+h/2*k1);
        k3 = -beta(t+h/2,x(i,:)+h/2)/lambda*(x(i,:)+h/2*k2-moreaudist
(t+h/2,x(i,:)+h/2*k2))-1/nu*(x(i,:)+h/2*k2...
            -moreau(t,x(i,:)+h/2*k2,x0,nu,fun_prueba_2))...
            +term_fuente(t+h/2,x(i,:)+h/2*k2);
        k4 = -beta(t+h,x(i,:)+h)/lambda*(x(i,:)+h*k3-moreaudist
(t+h,x(i,:)+h*k3))-1/nu*(x(i,:)+h*k3...
            -moreau(t,x(i,:)+h*k3,x0,nu,fun_prueba_2))...
            +term_fuente(t+h,x(i,:)+h*k3);
        x(i+1,:)=x(i,:)+h/6*(k1+2*(k2+k3)+k4);
        waitbar(i/m);
    end

elseif strcmp(class(dominiok),'function_handle')&&ischar(term_fuente)&&...
        strcmp(class(fun_prueba_2),'function_handle')
    for i=1:m
        t=T(i);
        [f,dist]=moreaudist(t,x(i,:));
        d(i)=dist;
        k1 = -beta(t,x(i,:))/lambda*(x(i,:)-f)...
            -1/nu*(x(i,:)-moreau(t,x(i,:),x0,nu,fun_prueba_2));
        k2 = -beta(t+h/2,x(i,:)+h/2)/lambda*(x(i,:)+h/2*k1-moreaudist
(t+h/2,x(i,:)+h/2*k1))-1/nu*(x(i,:)+h/2*k1...
```

```
            -moreau(t,x(i,:)+h/2*k1,x0,nu,fun_prueba_2));
        k3 = -beta(t+h/2,x(i,:)+h/2)/lambda*(x(i,:)+h/2*k2-moreaudist
(t+h/2,x(i,:)+h/2*k2))-1/nu*(x(i,:)+h/2*k2...
            -moreau(t,x(i,:)+h/2*k2,x0,nu,fun_prueba_2));
        k4 = -beta(t+h,x(i,:)+h)/lambda*(x(i,:)+h*k3-moreaudist
(t+h,x(i,:)+h*k3))-1/nu*(x(i,:)+h*k3...
            -moreau(t,x(i,:)+h*k3,x0,nu,fun_prueba_2));
        x(i+1,:)=x(i,:)+h/6*(k1+2*(k2+k3)+k4);
        waitbar(i/m);
    end

elseif strcmp(class(dominiok),'function_handle')&& strcmp
(class(term_fuente),'function_handle')&&ischar(fun_prueba_2)
    for i=1:m
        t=T(i);
        [f,dist]=moreaudist(t,x(i,:));
        d(i)=dist;
        k1 = -beta(t,x(i,:))/lambda*(x(i,:)-f)+term_fuente(t,x(i,:));
        k2 = -beta(t+h/2,x(i,:)+h/2)/lambda*(x(i,:)+h/2*k1-moreaudist
(t+h/2,x(i,:)+h/2*k1))...
            +term_fuente(t+h/2,x(i,:)+h/2*k1);
        k3 = -beta(t+h/2,x(i,:)+h/2)/lambda*(x(i,:)+h/2*k2-moreaudist
(t+h/2,x(i,:)+h/2*k2))...
            +term_fuente(t+h/2,x(i,:)+h/2*k2);
        k4 = -beta(t+h,x(i,:)+h)/lambda*(x(i,:)+h*k3-moreaudist
(t+h,x(i,:)+h*k3))...
            +term_fuente(t+h,x(i,:)+h*k3);
        x(i+1,:)=x(i,:)+h/6*(k1+2*(k2+k3)+k4);
        waitbar(i/m);
    end

elseif strcmp(class(term_fuente),'function_handle')&&...
        strcmp(class(fun_prueba_2),'function_handle')
    for i=1:m
        t=T(i);
        k1 = -1/lambda*(x(i,:)-moreau(t,x(i,:),x0,lambda,fun_prueba))...
            -beta(t,x(i,:))/nu*(x(i,:)-moreau(t,x(i,:),x0,nu,
fun_prueba_2))+term_fuente(t,x(i,:));
        k2 = -1/lambda*(x(i,:)+h/2*k1-moreau(t+h/2,x(i,:)+h/2*k1,...
            x0,lambda,fun_prueba))-beta(t+h/2,x(i,:)+h/2)/nu*(x(i,:)
+h/2*k1-moreau(t,x(i,:)+h/2*k1,x0,nu,fun_prueba_2))...
            +term_fuente(t+h/2,x(i,:)+h/2*k1);
        k3 = -1/lambda*(x(i,:)+h/2*k2-moreau(t+h/2,x(i,:)+h/2*k2,...
            x0,lambda,fun_prueba))-beta(t+h/2,x(i,:)+h/2)/nu*(x(i,:)
+h/2*k2-moreau(t,x(i,:)+h/2*k2,x0,nu,fun_prueba_2))...
            +term_fuente(t+h/2,x(i,:)+h/2*k2);
```

```
        k4 = -1/lambda*(x(i,:)+h*k3-moreau(t+h,x(i,:)+h*k3,...
            x0,lambda,fun_prueba))-beta(t+h,x(i,:)+h)/nu*(x(i,:)
+h*k3-moreau(t,x(i,:)+h*k3,x0,nu,fun_prueba_2))...
            +term_fuente(t+h,x(i,:)+h*k3);
        x(i+1,:)=x(i,:)+h/6*(k1+2*(k2+k3)+k4);
        waitbar(i/m);
    end

elseif strcmp(class(fun_prueba_2),'function_handle')&&...
        ischar(term_fuente)
    for i=1:m
        t=T(i);
        k1 = -1/lambda*(x(i,:)-moreau(t,x(i,:),x0,lambda,fun_prueba))...
            -beta(t)/nu*(x(i,:)-moreau(t,x(i,:),x0,nu,fun_prueba_2));
        k2 = -1/lambda*(x(i,:)+h/2*k1-moreau(t+h/2,x(i,:)+h/2*k1,...
            x0,lambda,fun_prueba))-beta(t+h/2,x(i,:)+h/2)/nu*(x(i,:)+h/2*k1...
            -moreau(t,x(i,:)+h/2*k1,x0,nu,fun_prueba_2));
        k3 = -1/lambda*(x(i,:)+h/2*k2-moreau(t+h/2,x(i,:)+h/2*k2,...
            x0,lambda,fun_prueba))-beta(t+h/2,x(i,:)+h/2)/nu*(x(i,:)+h/2*k2...
            -moreau(t,x(i,:)+h/2*k2,x0,nu,fun_prueba_2));
        k4 = -1/lambda*(x(i,:)+h*k3-moreau(t+h,x(i,:)+h*k3,...
            x0,lambda,fun_prueba))-beta(t+h,x(i,:)+h)/nu*(x(i,:)+h*k3...
            -moreau(t,x(i,:)+h*k3,x0,nu,fun_prueba_2));
        x(i+1,:)=x(i,:)+h/6*(k1+2*(k2+k3)+k4);
        waitbar(i/m);
    end

elseif strcmp(class(term_fuente),'function_handle')&&...
        ischar(fun_prueba_2)
    for i=1:m
        t=T(i);
        k1 = -1/lambda*(x(i,:)-moreau(t,x(i,:),x0,lambda,fun_prueba))...
            +term_fuente(t,x(i,:));
        k2 = -1/lambda*(x(i,:)+h/2*k1-moreau(t+h/2,x(i,:)+h/2*k1,...
            x0,lambda,fun_prueba))+term_fuente(t+h/2,x(i,:)+h/2*k1);
        k3 = -1/lambda*(x(i,:)+h/2*k2-moreau(t+h/2,x(i,:)+h/2*k2,...
            x0,lambda,fun_prueba))+term_fuente(t+h/2,x(i,:)+h/2*k2);
        k4 = -1/lambda*(x(i,:)+h*k3-moreau(t+h,x(i,:)+h*k3,...
            x0,lambda,fun_prueba))+term_fuente(t+h,x(i,:)+h*k3);
        x(i+1,:)=x(i,:)+h/6*(k1+2*(k2+k3)+k4);
        waitbar(i/m);
    end

elseif ischar(term_fuente)&& ischar(fun_prueba_2)
    for i=1:m
        t=T(i);
```

```
        k1 = -1/lambda*(x(i,:)-moreau(t,x(i,:),x0,lambda,fun_prueba));
        k2 = -1/lambda*(x(i,:)+h/2*k1-moreau(t+h/2,x(i,:)+h/2*k1,...
            x0,lambda,fun_prueba));
        k3 = -1/lambda*(x(i,:)+h/2*k2-moreau(t+h/2,x(i,:)+h/2*k2,...
            x0,lambda,fun_prueba));
        k4 = -1/lambda*(x(i,:)+h*k3-moreau(t+h,x(i,:)+h*k3,...
            x0,lambda,fun_prueba));
        x(i+1,:)=x(i,:)+h/6*(k1+2*(k2+k3)+k4);
        waitbar(i/m);
    end
end
close (u)
```

6.1.3 moreau.m

```
function f=moreau(t,x,x0,lambda,fun_prueba)
global dim

if dim==1
     f=fminsearch(@(y)(fun_prueba(t,y)+1/(2*lambda)*(y(1)-x(1))^2),x0);
elseif dim==2
     f=fminsearch(@(y)(fun_prueba(t,y)+1/(2*lambda)*((y(1)-x(1))^2+...
         (y(2)-x(2))^2)),x0);
elseif dim==3
     f=fminsearch(@(y)(fun_prueba(t,y)+1/(2*lambda)*((y(1)-x(1))^2+...
          (y(2)-x(2))^2+(y(3)-x(3))^2)),x0);
end
```

6.1.4 moreaudist.m

```
function [f,d]=moreaudist(t1,x)
global dim cx cy cz g1 g2 g3 t sol

if dim==1
    [g1]=centro(t1);
    cx=x(1);
    x0=[g1 cx];
    t=t1;
    [g,~]=solvopt(x0,'funcionobjetivo',[],[-1 1e-4 1e-6 15000 -1],
'funcionrestricciones');
    sol=x;
    [~,d]=solvopt(g1,'funciondistancia',[],[-1 1e-4 1e-6 15000 -1],
'funcionrestricciones');

elseif dim==2
    [g1,g2]=centro(t1);
```

```
     cx=x(1);
     cy=x(2);
     x0=[g1 g2 cx cy];
     t=t1;
     [g,~]=solvopt(x0,'funcionobjetivo',[],[-1 1e-4 1e-6 15000 -1],
'funcionrestricciones');
     sol=x;
     [~,d]=solvopt([g1 g2],'funciondistancia',[],[-1 1e-4 1e-6 15000 -1],
'funcionrestricciones');

elseif dim==3
    [g1,g2,g3]=centro(t1);
    cx=x(1);
    cy=x(2);
    cz=x(3);
    x0=[g1 g2 g3 cx cy cz];
    t=t1;
    [g,~]=solvopt(x0,'funcionobjetivo',[],[-1 1e-4 1e-6 15000 -1],
'funcionrestricciones');
    sol=x;
    [~,d]=solvopt([g1 g2 g3],'funciondistancia',[],[-1 1e-4 1e-6 15000 -1],
'funcionrestricciones');
end
f=g(:,dim+1:2*dim);
```

6.2 Interfaz gráfica

6.2.1 GUI.m

```
function varargout = GUI(varargin)
% GUI MATLAB code for GUI.fig
%      GUI, by itself, creates a new GUI or raises the existing
%      singleton*.
%
%      H = GUI returns the handle to a new GUI or the handle to
%      the existing singleton*.
%
%      GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI.M with the given input arguments.
%
%      GUI('Property','Value',...) creates a new GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
```

```
%      stop.  All inputs are passed to GUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI

% Last Modified by GUIDE v2.5 11-Jul-2014 19:03:29

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GUI is made visible.
function GUI_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;

guidata(hObject, handles);

axes(handles.formula_1);
formula = imread('imagenes\simple.png');
image(formula);
axis off
axis image
set(handles.seleccion_dimensiones,'Value',2);
set(handles.euler,'Value',1);

axes(handles.formula_2);
```

```
formula = imread('imagenes\distancia.png');
image(formula);
axis off
axis image
set(handles.seleccion_dimensiones_dist,'Value',2);
set(handles.euler_dist,'Value',1);
seleccion_dimensiones_dist_Callback(hObject, eventdata, handles)
set(handles.menu_front,'Value',1)
menu_front_Callback(hObject, eventdata, handles)

function varargout = GUI_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;


function bip_comp_Callback(hObject, eventdata, handles)

if get(handles.bip_comp,'Value')==1
    if get(handles.fuen_comp,'Value')==1
        axes(handles.formula_1)
        formula = imread('imagenes\bipotencial_term_fuente.png');
        image(formula);
        axis off
        axis image
        set(handles.bip_int_1,'Enable','on');set(handles.bip_int_2,
'Enable','on');set(handles.bip_int_3,'Enable','on');
        set(handles.nu_1,'Enable','on');set(handles.nu_2,'Enable','on');
set(handles.nu_3,'Enable','on');set(handles.nu_4,'Enable','on');
        set(handles.beta_1,'Enable','on');set(handles.beta_2,
'Enable','on');set(handles.beta_3,'Enable','on');
        set(handles.fuen_int_1,'Enable','on');set(handles.fuen_int_2,
'Enable','on');set(handles.fuen_int_3,'Enable','on');
    else
        axes(handles.formula_1)
        formula = imread('imagenes\bipotencial.png');
        image(formula);
        axis off
        axis image
        set(handles.bip_int_1,'Enable','on');set(handles.bip_int_2,
'Enable','on');set(handles.bip_int_3,'Enable','on');
        set(handles.nu_1,'Enable','on');set(handles.nu_2,'Enable','on');
set(handles.nu_3,'Enable','on');set(handles.nu_4,'Enable','on');
        set(handles.beta_1,'Enable','on');set(handles.beta_2,
'Enable','on');set(handles.beta_3,'Enable','on');
        set(handles.fuen_int_1,'Enable','off');set(handles.fuen_int_2,
'Enable','off');set(handles.fuen_int_3,'Enable','off');
    end
```

```
elseif get(handles.fuen_comp,'Value')==1
    axes(handles.formula_1)
    formula = imread('imagenes\term_fuente.png');
    image(formula);
    axis off
    axis image
    set(handles.bip_int_1,'Enable','off');set(handles.bip_int_2,'Enable',
'off');set(handles.bip_int_3,'Enable','off');
    set(handles.nu_1,'Enable','off');set(handles.nu_2,'Enable','off');
set(handles.nu_3,'Enable','off');set(handles.nu_4,'Enable','off');
    set(handles.beta_1,'Enable','off');set(handles.beta_2,
'Enable','off');
set(handles.beta_3,'Enable','off');
    set(handles.fuen_int_1,'Enable','on');set(handles.fuen_int_2,
'Enable','on');set(handles.fuen_int_3,'Enable','on');
else
    axes(handles.formula_1)
    formula = imread('imagenes\simple.png');
    image(formula);
    axis off
    axis image
    set(handles.bip_int_1,'Enable','off');set(handles.bip_int_2,
'Enable','off');set(handles.bip_int_3,'Enable','off');
    set(handles.nu_1,'Enable','off');set(handles.nu_2,'Enable','off');
set(handles.nu_3,'Enable','off');set(handles.nu_4,'Enable','off');
    set(handles.beta_1,'Enable','off');set(handles.beta_2,
'Enable','off');set(handles.beta_3,'Enable','off');
    set(handles.fuen_int_1,'Enable','off');set(handles.fuen_int_2,
'Enable','off');set(handles.fuen_int_3,'Enable','off');
end


function fuen_comp_Callback(hObject, eventdata, handles)

if get(handles.bip_comp,'Value')==1
    if get(handles.fuen_comp,'Value')==1
        axes(handles.formula_1)
        formula = imread('imagenes\bipotencial_term_fuente.png');
        image(formula);
        axis off
        axis image
        set(handles.bip_int_1,'Enable','on');set(handles.bip_int_2,
'Enable','on');set(handles.bip_int_3,'Enable','on');
        set(handles.nu_1,'Enable','on');set(handles.nu_2,
'Enable','on');set(handles.nu_3,'Enable','on');
set(handles.nu_4,'Enable','on');
```

```
        set(handles.beta_1,'Enable','on');set(handles.beta_2,
'Enable','on');set(handles.beta_3,'Enable','on');
        set(handles.fuen_int_1,'Enable','on');set(handles.fuen_int_2,
'Enable','on');set(handles.fuen_int_3,'Enable','on');
    else
        axes(handles.formula_1)
        formula = imread('imagenes\bipotencial.png');
        image(formula);
        axis off
        axis image
        set(handles.bip_int_1,'Enable','on');set(handles.bip_int_2,
'Enable','on');set(handles.bip_int_3,'Enable','on');
        set(handles.nu_1,'Enable','on');set(handles.nu_2,
'Enable','on');set(handles.nu_3,'Enable','on');
set(handles.nu_4,'Enable','on');
        set(handles.beta_1,'Enable','on');set(handles.beta_2,
'Enable','on');set(handles.beta_3,'Enable','on');
        set(handles.fuen_int_1,'Enable','off');set(handles.fuen_int_2,
'Enable','off');set(handles.fuen_int_3,'Enable','off');
    end
elseif get(handles.fuen_comp,'Value')==1
    axes(handles.formula_1)
    formula = imread('imagenes\term_fuente.png');
    image(formula);
    axis off
    axis image
    set(handles.bip_int_1,'Enable','off');set(handles.bip_int_2,
'Enable','off');set(handles.bip_int_3,'Enable','off');
    set(handles.nu_1,'Enable','off');set(handles.nu_2,'Enable','off');
set(handles.nu_3,'Enable','off');set(handles.nu_4,'Enable','off');
    set(handles.beta_1,'Enable','off');set(handles.beta_2,
'Enable','off');set(handles.beta_3,'Enable','off');
    set(handles.fuen_int_1,'Enable','on');set(handles.fuen_int_2,
'Enable','on');set(handles.fuen_int_3,'Enable','on');
else
    axes(handles.formula_1)
    formula = imread('imagenes\simple.png');
    image(formula);
    axis off
    axis image
    set(handles.bip_int_1,'Enable','off');set(handles.bip_int_2,'Enable',
'off');set(handles.bip_int_3,'Enable','off');
    set(handles.nu_1,'Enable','off');set(handles.nu_2,'Enable','off');
set(handles.nu_3,'Enable','off');set(handles.nu_4,'Enable','off');
    set(handles.beta_1,'Enable','off');set(handles.beta_2,'Enable','off');
set(handles.beta_3,'Enable','off');
```

```
    set(handles.fuen_int_1,'Enable','off');set(handles.fuen_int_2,
'Enable','off');set(handles.fuen_int_3,'Enable','off');
end

function fun_prueba_Callback(hObject, eventdata, handles)

function fun_prueba_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function fuen_int_1_Callback(hObject, eventdata, handles)

function fuen_int_1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bip_int_1_Callback(hObject, eventdata, handles)

function bip_int_1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function seleccion_dimensiones_Callback(hObject, eventdata, handles)

function seleccion_dimensiones_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function euler_Callback(hObject, eventdata, handles)

function rk4_Callback(hObject, eventdata, handles)
```

```
function a_Callback(hObject, eventdata, handles)

function a_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function b_Callback(hObject, eventdata, handles)

function b_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function m_Callback(hObject, eventdata, handles)

function m_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function lambda_Callback(hObject, eventdata, handles)

function lambda_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function nu_1_Callback(hObject, eventdata, handles)

function nu_1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function beta_1_Callback(hObject, eventdata, handles)
```

```
function beta_1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function x0_Callback(hObject, eventdata, handles)

function x0_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function animacion_Callback(hObject, eventdata, handles)

if get(handles.animacion,'Value')==1
    set(handles.p_1,'Enable','on');set(handles.p_2,'Enable','on');
else
    set(handles.p_1,'Enable','off');set(handles.p_2,'Enable','off');
end

function p_1_Callback(hObject, eventdata, handles)

function p_1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
 get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function calcular_Callback(hObject, eventdata, handles)
global dim x0 a b m lambda nu texto_fun_prueba texto_fun_prueba_2 ...
    texto_term_fuente texto_beta x y tipo_calculo p

tipo_calculo=1;
j=get(handles.fun_prueba,'String');
texto_fun_prueba=str2func(strcat('@(t,x)',j));
k=get(handles.bip_int_1,'String');
texto_fun_prueba_2=str2func(strcat('@(t,x)',k));
l=get(handles.fuen_int_1,'String');
texto_term_fuente=str2func(strcat('@(t,x)',l));
n=get(handles.beta_1,'String');
texto_beta=str2func(strcat('@(t,x)',n));
```

```
fig_resultados= axes('Parent',handles.panel_resultados,'Position',
[.1 .1 .9 .9]);
dim=get(handles.seleccion_dimensiones,'Value');
x0=str2num(get(handles.x0,'String'));
a=str2double(get(handles.a,'String'));
b=str2double(get(handles.b,'String'));
m=str2double(get(handles.m,'String'));
lambda=str2double(get(handles.lambda,'String'));
nu=str2double(get(handles.nu_1,'String'));
p=str2double(get(handles.p_1,'String'));


if get(handles.euler,'Value')==1&&get(handles.rk4,'Value')==0
    if get(handles.fuen_comp,'Value')==0&&get(handles.bip_comp,
'Value')==0
        x=euler(@fun_prueba,x0,a,b,m,lambda,'','','','','');
    elseif get(handles.fuen_comp,'Value')==1&&get(handles.bip_comp,
'Value')==0
        x=euler(@fun_prueba,x0,a,b,m,lambda,@term_fuente,'','','','');
    elseif get(handles.fuen_comp,'Value')==0&&get(handles.bip_comp,
'Value')==1
        x=euler(@fun_prueba,x0,a,b,m,lambda,'',@fun_prueba_2,
nu,@beta,'');
    else
        x=euler(@fun_prueba,x0,a,b,m,lambda,@term_fuente,
@fun_prueba_2,nu,@beta,'');
    end
    set(handles.panel_datos,'Visible','off')
    set(handles.panel_resultados,'Visible','on')
    if get(handles.animacion,'Value')==1
        set(handles.panel_resultados,'Visible','on')
        subplot(1,1,1); plot(0,0)
        animacion(x,p)
    else
        set(handles.panel_resultados,'Visible','on')
        subplot(1,1,1); plot(0,0)
        dibujar(x)
    end

elseif get(handles.euler,'Value')==0&&get(handles.rk4,'Value')==1
    if get(handles.fuen_comp,'Value')==0&&get(handles.bip_comp,
'Value')==0
        x=rk4(@fun_prueba,x0,a,b,m,lambda,'','','','','');
    elseif get(handles.fuen_comp,'Value')==1&&get(handles.bip_comp,
'Value')==0
        x=rk4(@fun_prueba,x0,a,b,m,lambda,@term_fuente,'','','','');
```

```
    elseif get(handles.fuen_comp,'Value')==0&&get(handles.bip_comp,
'Value')==1
        x=rk4(@fun_prueba,x0,a,b,m,lambda,'',@fun_prueba_2,
nu,@beta,'');
    else
        x=rk4(@fun_prueba,x0,a,b,m,lambda,@term_fuente,
@fun_prueba_2,nu,@beta,'');
    end
    set(handles.panel_datos,'Visible','off')
    set(handles.panel_resultados,'Visible','on')
    if get(handles.animacion,'Value')==1
        set(handles.panel_resultados,'Visible','on')
        subplot(1,1,1); plot(0,0)
        animacion(x,p)
    else
        set(handles.panel_resultados,'Visible','on')
        subplot(1,1,1); plot(0,0)
        dibujar(x)
    end

elseif get(handles.euler,'Value')==1&&get(handles.rk4,'Value')==1
    if get(handles.fuen_comp,'Value')==0&&get(handles.bip_comp,
'Value')==0
        x=euler(@fun_prueba,x0,a,b,m,lambda,'','','','','');
        y=rk4(@fun_prueba,x0,a,b,m,lambda,'','','','','');
    elseif get(handles.fuen_comp,'Value')==1&&get(handles.bip_comp,
'Value')==0
        x=euler(@fun_prueba,x0,a,b,m,lambda,@term_fuente,'','','','');
        y=rk4(@fun_prueba,x0,a,b,m,lambda,@term_fuente,'','','','');
    elseif get(handles.fuen_comp,'Value')==0&&get(handles.bip_comp,
'Value')==1
        x=euler(@fun_prueba,x0,a,b,m,lambda,'',@fun_prueba_2,
nu,@beta,'');
        y=rk4(@fun_prueba,x0,a,b,m,lambda,'',@fun_prueba_2,
nu,@beta,'');
    else
        x=euler(@fun_prueba,x0,a,b,m,lambda,@term_fuente,
@fun_prueba_2,nu,@beta,'');
        y=rk4(@fun_prueba,x0,a,b,m,lambda,@term_fuente,
@fun_prueba_2,nu,@beta,'');
    end
    set(handles.panel_datos,'Visible','off')
    set(handles.panel_resultados,'Visible','on')
    if get(handles.animacion,'Value')==1
        set(handles.panel_resultados,'Visible','on')
        subplot(1,1,1); plot(0,0)
```

```
        animacionh(x,y,'Euler','Runge-Kutta 4',p)
    else
        set(handles.panel_resultados,'Visible','on')
        subplot(1,1,1); plot(0,0)
        comparacionh(x,y,'Euler','Runge-Kutta 4')
    end

end
set(handles.repetir,'Enable','on');
set(handles.exportar,'Enable','on');
set(handles.cargar_datos,'Enable','on');

function seleccion_dimensiones_dist_Callback(hObject, eventdata, handles)
if get(handles.menu_front,'Value')==1
    if  get(handles.seleccion_dimensiones_dist,'Value')==1
        set(handles.x_param_1,'Enable','on');set(handles.x_param_2,
'Enable','on');
        set(handles.y_param_1,'Enable','off');set(handles.y_param_2,
'Enable','off');
        set(handles.z_param_1,'Enable','off');set(handles.z_param_2,
'Enable','off');
        set(handles.x_centro_1,'Enable','on');set(handles.x_centro_2,
'Enable','on');
        set(handles.y_centro_1,'Enable','off');set(handles.y_centro_2,
'Enable','off');
        set(handles.z_centro_1,'Enable','off');set(handles.z_centro_2,
'Enable','off');
        set(handles.i_def,'Enable','on');set(handles.i_inf,'Enable','on');
set(handles.i_sup,'Enable','on');
        set(handles.j_def,'Enable','off');set(handles.j_inf,'Enable','off');
set(handles.j_sup,'Enable','off');
    elseif get(handles.seleccion_dimensiones_dist,'Value')==2
        set(handles.x_param_1,'Enable','on');set(handles.x_param_2,
'Enable','on');
        set(handles.y_param_1,'Enable','on');set(handles.y_param_2,
'Enable','on');
        set(handles.z_param_1,'Enable','off');set(handles.z_param_2,
'Enable','off');
        set(handles.x_centro_1,'Enable','on');set(handles.x_centro_2,
'Enable','on');
        set(handles.y_centro_1,'Enable','on');set(handles.y_centro_2,
'Enable','on');
        set(handles.z_centro_1,'Enable','off');set(handles.z_centro_2,
'Enable','off');
        set(handles.i_def,'Enable','on');set(handles.i_inf,'Enable','on');
set(handles.i_sup,'Enable','on');
```

```
        set(handles.j_def,'Enable','off');set(handles.j_inf,'Enable','off');
set(handles.j_sup,'Enable','off');
    else
        set(handles.x_param_1,'Enable','on');set(handles.x_param_2,
'Enable','on');
        set(handles.y_param_1,'Enable','on');set(handles.y_param_2,
'Enable','on');
        set(handles.z_param_1,'Enable','on');set(handles.z_param_2,
'Enable','on');
        set(handles.x_centro_1,'Enable','on');set(handles.x_centro_2,
'Enable','on');
        set(handles.y_centro_1,'Enable','on');set(handles.y_centro_2,
'Enable','on');
        set(handles.z_centro_1,'Enable','on');set(handles.z_centro_2,
'Enable','on');set(handles.i_def,'Enable','on');
set(handles.i_inf,'Enable','on');set(handles.i_sup,'Enable','on');
        set(handles.j_def,'Enable','on');set(handles.j_inf,'Enable','on');
set(handles.j_sup,'Enable','on');
    end
elseif get(handles.menu_front,'Value')==2
    if  get(handles.seleccion_dimensiones_dist,'Value')==1
        set(handles.x_cart_1,'Enable','on');set(handles.x_cart_2,
'Enable','on');set(handles.x_cart_3,'Enable','on');
        set(handles.y_cart_1,'Enable','off');set(handles.y_cart_2,
'Enable','off');set(handles.y_cart_3,'Enable','off');
        set(handles.z_cart_1,'Enable','off');set(handles.z_cart_2,
'Enable','off');set(handles.z_cart_3,'Enable','off');
        set(handles.x_centro_1,'Enable','on');set(handles.x_centro_2,
'Enable','on');
        set(handles.y_centro_1,'Enable','off');set(handles.y_centro_2,
'Enable','off');
        set(handles.z_centro_1,'Enable','off');set(handles.z_centro_2,
'Enable','off');
    elseif get(handles.seleccion_dimensiones_dist,'Value')==2
        set(handles.x_cart_1,'Enable','on');set(handles.x_cart_2,
'Enable','on');set(handles.x_cart_3,'Enable','on');
        set(handles.y_cart_1,'Enable','on');set(handles.y_cart_2,
'Enable','on');set(handles.y_cart_3,'Enable','on');
        set(handles.z_cart_1,'Enable','off');set(handles.z_cart_2,
'Enable','off');set(handles.z_cart_3,'Enable','off');
        set(handles.x_centro_1,'Enable','on');set(handles.x_centro_2,
'Enable','on');
        set(handles.y_centro_1,'Enable','on');set(handles.y_centro_2,
'Enable','on');
        set(handles.z_centro_1,'Enable','off');set(handles.z_centro_2,
'Enable','off');
```

```
    else
        set(handles.x_cart_1,'Enable','on');set(handles.x_cart_2,
'Enable','on');set(handles.x_cart_3,'Enable','on');
        set(handles.y_cart_1,'Enable','on');set(handles.y_cart_2,
'Enable','on');set(handles.y_cart_3,'Enable','on');
        set(handles.z_cart_1,'Enable','on');set(handles.z_cart_2,
'Enable','on');set(handles.z_cart_3,'Enable','on');
        set(handles.x_centro_1,'Enable','on');set(handles.x_centro_2,
'Enable','on');
        set(handles.y_centro_1,'Enable','on');set(handles.y_centro_2,
'Enable','on');
        set(handles.z_centro_1,'Enable','on');set(handles.z_centro_2,
'Enable','on');
    end
end

function seleccion_dimensiones_dist_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function calcular_dist_Callback(hObject, eventdata, handles)
global dim x0 a b m lambda nu texto_funcionrestricciones...
    x_centro y_centro z_centro x_param y_param z_param...
    x_cart_1 x_cart_2 y_cart_1 y_cart_2 z_cart_1 z_cart_2...
    x y d d1 d2 X Y Z p1 p2 i_inf i_sup j_inf j_sup tam tam_param...
    texto_fun_prueba_2 texto_term_fuente texto_beta tipo_calculo

tipo_calculo=2;
j=get(handles.int_dominiok,'String');
texto_funcionrestricciones=str2func(strcat('@(t,x)',j));
e=get(handles.x_centro_1,'String');
x_centro=str2func(strcat('@(t)',e));
r=get(handles.y_centro_1,'String');
y_centro=str2func(strcat('@(t)',r));
u=get(handles.z_centro_1,'String');
z_centro=str2func(strcat('@(t)',u));
x_param=get(handles.x_param_1,'String');
y_param=get(handles.y_param_1,'String');
z_param=get(handles.z_param_1,'String');
x_cart_1=get(handles.x_cart_1,'String');
x_cart_2=get(handles.x_cart_2,'String');
y_cart_1=get(handles.y_cart_1,'String');
y_cart_2=get(handles.y_cart_2,'String');
```

```
z_cart_1=get(handles.z_cart_1,'String');
z_cart_2=get(handles.z_cart_2,'String');
i_inf=str2double(get(handles.i_inf,'String'));
i_sup=str2double(get(handles.i_sup,'String'));
j_inf=str2double(get(handles.j_inf,'String'));
j_sup=str2double(get(handles.j_sup,'String'));
tam=str2double(get(handles.tam,'String'));
tam_param=str2double(get(handles.tam_param,'String'));

fig_resultados= axes('Parent',handles.panel_resultados,'Position',
[.1 .1 .9 .9],'Color','b');
dim=get(handles.seleccion_dimensiones_dist,'Value');
x0=str2num(get(handles.x0_dist,'String'));
a=str2double(get(handles.a_dist,'String'));
b=str2double(get(handles.b_dist,'String'));
m=str2double(get(handles.m_dist,'String'));
lambda=str2double(get(handles.lambda_dist,'String'));
nu=str2double(get(handles.nu_1_dist,'String'));
p1=str2double(get(handles.p1_1_dist,'String'));
p2=str2double(get(handles.p2_1_dist,'String'));
k=get(handles.bip_int_1_dist,'String');
texto_fun_prueba_2=str2func(strcat('@(t,x)',k));
l=get(handles.fuen_int_1_dist,'String');
texto_term_fuente=str2func(strcat('@(t,x)',l));
n=get(handles.beta_1_dist,'String');
texto_beta=str2func(strcat('@(t,x)',n));

if get(handles.euler_dist,'Value')==1&&get(handles.rk4_dist,'Value')==0
    if get(handles.fuen_comp_dist,'Value')==0&&get(handles.bip_comp_dist,
'Value')==0
        [x,d]=euler(@fun_prueba,x0,a,b,m,lambda,'','','','',@dominiok);
    elseif get(handles.fuen_comp_dist,'Value')==1&&get
(handles.bip_comp_dist,'Value')==0
        [x,d]=euler(@fun_prueba,x0,a,b,m,lambda,@term_fuente,'','','',
@dominiok);
    elseif get(handles.fuen_comp_dist,'Value')==0&&get
(handles.bip_comp_dist,'Value')==1
        [x,d]=euler(@fun_prueba,x0,a,b,m,lambda,'',@fun_prueba_2,
nu,@beta,@dominiok);
    else
        [x,d]=euler(@fun_prueba,x0,a,b,m,lambda,@term_fuente,
@fun_prueba_2,nu,@beta,'');
    end

elseif get(handles.euler_dist,'Value')==0&&get(handles.rk4_dist,'Value')==1
    if get(handles.fuen_comp_dist,'Value')==0&&get
```

```
(handles.bip_comp_dist,'Value')==0
        [x,d]=rk4(@fun_prueba,x0,a,b,m,lambda,'','','','',@dominiok);
    elseif get(handles.fuen_comp_dist,'Value')==1&&get
(handles.bip_comp_dist,'Value')==0
        [x,d]=rk4(@fun_prueba,x0,a,b,m,lambda,@term_fuente,'','','',@dominiok);
    elseif get(handles.fuen_comp_dist,'Value')==0&&get
(handles.bip_comp_dist,'Value')==1
        [x,d]=rk4(@fun_prueba,x0,a,b,m,lambda,'',@fun_prueba_2,
nu,@beta,@dominiok);
    else
        [x,d]=rk4(@fun_prueba,x0,a,b,m,lambda,@term_fuente,
@fun_prueba_2,nu,@beta,'');
    end
end

if get(handles.euler_dist,'Value')==1&&get(handles.rk4_dist,'Value')==0||...
        get(handles.euler_dist,'Value')==0&&get(handles.rk4_dist,'Value')==1
set(handles.panel_distancia,'Visible','off')
    if get(handles.animacion_dist,'Value')==1&&get
(handles.tipo_animacion_1,'Value')==1
        [X,Y,Z]=dominio(get(handles.menu_front,'Value'));
        set(handles.panel_resultados,'Visible','on')
        subplot(1,1,1); plot(0,0)
        animacion1(x,d,X,Y,Z,p1,p2)
    elseif get(handles.animacion_dist,'Value')==1&&get
(handles.tipo_animacion_1,'Value')==2
        [X,Y,Z]=dominio(get(handles.menu_front,'Value'));
        set(handles.panel_resultados,'Visible','on')
        subplot(1,1,1); plot(0,0)
        animacion2(x,d,X,Y,Z,p1,p2)
    elseif get(handles.animacion_dist,'Value')==1&&get
(handles.tipo_animacion_1,'Value')==3
        [X,Y,Z]=dominio(get(handles.menu_front,'Value'));
        set(handles.panel_resultados,'Visible','on')
        subplot(1,1,1); plot(0,0)
        animacion3(x,d,X,Y,Z,p1,p2)
    else
        set(handles.panel_resultados,'Visible','on')
        subplot(1,1,1); plot(0,0)
        dibujar_dist(x,d)
    end
end

if get(handles.euler_dist,'Value')==1&&get(handles.rk4_dist,'Value')==1
    if get(handles.fuen_comp_dist,'Value')==0&&get
(handles.bip_comp_dist,'Value')==0
```

```
        [x,d1]=euler(@fun_prueba,x0,a,b,m,lambda,'','','','',@dominiok);
        [y,d2]=rk4(@fun_prueba,x0,a,b,m,lambda,'','','','',@dominiok);
    elseif get(handles.fuen_comp_dist,'Value')==1&&get
(handles.bip_comp_dist,'Value')==0
        [x,d1]=euler(@fun_prueba,x0,a,b,m,lambda,@term_fuente,
'','','',@dominiok);
        [y,d2]=rk4(@fun_prueba,x0,a,b,m,lambda,@term_fuente,
'','','',@dominiok);
    elseif get(handles.fuen_comp_dist,'Value')==0&&get
(handles.bip_comp_dist,'Value')==1
        [x,d1]=euler(@fun_prueba,x0,a,b,m,lambda,'',@fun_prueba_2,nu,
@beta,@dominiok);
        [y,d2]=rk4(@fun_prueba,x0,a,b,m,lambda,'',@fun_prueba_2,nu,
@beta,@dominiok);
    else
        [x,d1]=rk4(@fun_prueba,x0,a,b,m,lambda,@term_fuente,
@fun_prueba_2,nu,@beta,@dominiok);
        [y,d2]=rk4(@fun_prueba,x0,a,b,m,lambda,@term_fuente,
@fun_prueba_2,nu,@beta,@dominiok);
    end
    set(handles.panel_distancia,'Visible','off')
    set(handles.panel_resultados,'Visible','on')
    subplot(1,1,1); plot(0,0)
    comparacionh_dist(x,y,d1,d2,'Euler','Runge-Kutta 4')

end
set(handles.repetir,'Enable','on');
set(handles.exportar,'Enable','on');
set(handles.cargar_datos,'Enable','on');

function euler_dist_Callback(hObject, eventdata, handles)

if get(handles.rk4_dist,'Value')==1&&get(handles.euler_dist,'Value')==1
    set(handles.animacion_dist,'Enable','off');
    set(handles.p1_1_dist,'Enable','off');set(handles.p1_2_dist,
'Enable','off');
    set(handles.p2_1_dist,'Enable','off');set(handles.p2_2_dist,
'Enable','off');
    set(handles.tipo_animacion_1,'Enable','off');set
(handles.tipo_animacion_2,'Enable','off');
else
    set(handles.animacion_dist,'Enable','on');
    set(handles.p1_1_dist,'Enable','on');set(handles.p1_2_dist,
'Enable','on');
    set(handles.p2_1_dist,'Enable','on');set(handles.p2_2_dist,
'Enable','on');
```

```
    set(handles.tipo_animacion_1,'Enable','on');set
(handles.tipo_animacion_2,'Enable','on');
end

function rk4_dist_Callback(hObject, eventdata, handles)

if get(handles.rk4_dist,'Value')==1&&get(handles.euler_dist,'Value')==1
    set(handles.animacion_dist,'Enable','off');
    set(handles.p1_1_dist,'Enable','off');set(handles.p1_2_dist,
'Enable','off');
    set(handles.p2_1_dist,'Enable','off');set(handles.p2_2_dist,
'Enable','off');
    set(handles.tipo_animacion_1,'Enable','off');set
(handles.tipo_animacion_2,'Enable','off');
else
    set(handles.animacion_dist,'Enable','on');
    set(handles.p1_1_dist,'Enable','on');set(handles.p1_2_dist,
'Enable','on');
    set(handles.p2_1_dist,'Enable','on');set(handles.p2_2_dist,
'Enable','on');
    set(handles.tipo_animacion_1,'Enable','on');set
(handles.tipo_animacion_2,'Enable','on');
end

function a_dist_Callback(hObject, eventdata, handles)

function a_dist_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function b_dist_Callback(hObject, eventdata, handles)

function b_dist_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function m_dist_Callback(hObject, eventdata, handles)

function m_dist_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function lambda_dist_Callback(hObject, eventdata, handles)

function lambda_dist_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function x0_dist_Callback(hObject, eventdata, handles)

function x0_dist_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function animacion_dist_Callback(hObject, eventdata, handles)
if get(handles.animacion_dist,'Value')==1
    set(handles.p1_1_dist,'Enable','on');set(handles.p1_2_dist,
'Enable','on');
    set(handles.p2_1_dist,'Enable','on');set(handles.p2_2_dist,
'Enable','on');
    set(handles.tipo_animacion_1,'Enable','on');set
(handles.tipo_animacion_2,'Enable','on');
else
    set(handles.p1_1_dist,'Enable','off');set(handles.p1_2_dist,
'Enable','off');
    set(handles.p2_1_dist,'Enable','off');set(handles.p2_2_dist,
'Enable','off');
    set(handles.tipo_animacion_1,'Enable','off');set
(handles.tipo_animacion_2,'Enable','off');
end

function p1_1_dist_Callback(hObject, eventdata, handles)

function p1_1_dist_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end

function p2_1_dist_Callback(hObject, eventdata, handles)

function p2_1_dist_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function int_dominiok_Callback(hObject, eventdata, handles)

function int_dominiok_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
 get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function menu_Callback(hObject, eventdata, handles)

function datos_Callback(hObject, eventdata, handles)

function resultados_Callback(hObject, eventdata, handles)
set(handles.texto1,'Visible','off');set(handles.texto2,
'Visible','off');set(handles.texto3,'Visible','off');
set(handles.panel_datos,'Visible','off')
set(handles.panel_distancia,'Visible','off')
set(handles.panel_resultados,'Visible','on')
set(handles.cargar_datos,'Enable','on')


function general_Callback(hObject, eventdata, handles)
set(handles.texto1,'Visible','off');set(handles.texto2,
'Visible','off');set(handles.texto3,'Visible','off');
set(handles.panel_resultados,'Visible','off')
set(handles.panel_distancia,'Visible','off')
set(handles.panel_datos,'Visible','on')
set(handles.cargar_datos,'Enable','off')

function ejemplos_Callback(hObject, eventdata, handles)

function distancia_Callback(hObject, eventdata, handles)
set(handles.texto1,'Visible','off');set(handles.texto2,
'Visible','off');set(handles.texto3,'Visible','off');
```

```
set(handles.panel_resultados,'Visible','off')
set(handles.panel_datos,'Visible','off')
set(handles.panel_distancia,'Visible','on')
set(handles.cargar_datos,'Enable','off')

function tipo_animacion_1_Callback(hObject, eventdata, handles)

function tipo_animacion_1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function bip_comp_dist_Callback(hObject, eventdata, handles)

if get(handles.bip_comp_dist,'Value')==1
    if get(handles.fuen_comp_dist,'Value')==1
        axes(handles.formula_2)
        formula = imread('imagenes\bipotencial_term_fuente_dist.png');
        image(formula);
        axis off
        axis image
        set(handles.bip_int_1_dist,'Enable','on');set
(handles.bip_int_2_dist,'Enable','on');
set(handles.bip_int_3_dist,'Enable','on');
        set(handles.nu_1_dist,'Enable','on');
set(handles.nu_2_dist,'Enable','on');
set(handles.nu_3_dist,'Enable','on');
set(handles.nu_4_dist,'Enable','on');
        set(handles.beta_1_dist,'Enable','on');
set(handles.beta_2_dist,'Enable','on');
set(handles.beta_3_dist,'Enable','on');
        set(handles.fuen_int_1_dist,'Enable','on');
set(handles.fuen_int_2_dist,'Enable','on');
set(handles.fuen_int_3_dist,'Enable','on');
    else
        axes(handles.formula_2)
        formula = imread('imagenes\bipotencial_dist.png');
        image(formula);
        axis off
        axis image
        set(handles.bip_int_1_dist,'Enable','on');
set(handles.bip_int_2_dist,'Enable','on');
set(handles.bip_int_3_dist,'Enable','on');
```

```
        set(handles.nu_1_dist,'Enable','on');
set(handles.nu_2_dist,'Enable','on');
set(handles.nu_3_dist,'Enable','on');
set(handles.nu_4_dist,'Enable','on');
        set(handles.beta_1_dist,'Enable','on');
set(handles.beta_2_dist,'Enable','on');
set(handles.beta_3_dist,'Enable','on');
        set(handles.fuen_int_1_dist,'Enable','off');
set(handles.fuen_int_2_dist,'Enable','off');
set(handles.fuen_int_3_dist,'Enable','off');
    end
elseif get(handles.fuen_comp_dist,'Value')==1
    axes(handles.formula_2)
    formula = imread('imagenes\term_fuente_dist.png');
    image(formula);
    axis off
    axis image
    set(handles.bip_int_1_dist,'Enable','off');
set(handles.bip_int_2_dist,'Enable','off');
set(handles.bip_int_3_dist,'Enable','off');
    set(handles.nu_1_dist,'Enable','off');
set(handles.nu_2_dist,'Enable','off');
set(handles.nu_3_dist,'Enable','off');
set(handles.nu_4_dist,'Enable','off');
    set(handles.beta_1_dist,'Enable','on');
set(handles.beta_2_dist,'Enable','on');
set(handles.beta_3_dist,'Enable','on');
    set(handles.fuen_int_1_dist,'Enable','on');
set(handles.fuen_int_2_dist,'Enable','on');
set(handles.fuen_int_3_dist,'Enable','on');
else
    axes(handles.formula_2)
    formula = imread('imagenes\distancia.png');
    image(formula);
    axis off
    axis image
    set(handles.bip_int_1_dist,'Enable','off');
set(handles.bip_int_2_dist,'Enable','off');
set(handles.bip_int_3_dist,'Enable','off');
    set(handles.nu_1_dist,'Enable','off');
set(handles.nu_2_dist,'Enable','off');
set(handles.nu_3_dist,'Enable','off');
set(handles.nu_4_dist,'Enable','off');
    set(handles.beta_1_dist,'Enable','off');
set(handles.beta_2_dist,'Enable','off');
set(handles.beta_3_dist,'Enable','off');
```

```
    set(handles.fuen_int_1_dist,'Enable','off');
set(handles.fuen_int_2_dist,'Enable','off');
set(handles.fuen_int_3_dist,'Enable','off');
end


function fuen_comp_dist_Callback(hObject, eventdata, handles)

if get(handles.bip_comp_dist,'Value')==1
    if get(handles.fuen_comp_dist,'Value')==1
        axes(handles.formula_2)
        formula = imread
('imagenes\bipotencial_term_fuente_dist.png');
        image(formula);
        axis off
        axis image
        set(handles.bip_int_1_dist,'Enable','on');
set(handles.bip_int_2_dist,'Enable','on');
set(handles.bip_int_3_dist,'Enable','on');
        set(handles.nu_1_dist,'Enable','on');
set(handles.nu_2_dist,'Enable','on');
set(handles.nu_3_dist,'Enable','on');
set(handles.nu_4_dist,'Enable','on');
        set(handles.beta_1_dist,'Enable','on');
set(handles.beta_2_dist,'Enable','on');
set(handles.beta_3_dist,'Enable','on');
        set(handles.fuen_int_1_dist,'Enable','on');
set(handles.fuen_int_2_dist,'Enable','on');
set(handles.fuen_int_3_dist,'Enable','on');
    else
        axes(handles.formula_2)
        formula = imread('imagenes\bipotencial_dist.png');
        image(formula);
        axis off
        axis image
        set(handles.bip_int_1_dist,'Enable','on');
set(handles.bip_int_2_dist,'Enable','on');
set(handles.bip_int_3_dist,'Enable','on');
        set(handles.nu_1_dist,'Enable','on');
set(handles.nu_2_dist,'Enable','on');
set(handles.nu_3_dist,'Enable','on');
set(handles.nu_4_dist,'Enable','on');
        set(handles.beta_1_dist,'Enable','on');
set(handles.beta_2_dist,'Enable','on');
set(handles.beta_3_dist,'Enable','on');
        set(handles.fuen_int_1_dist,'Enable','off');
```

```
set(handles.fuen_int_2_dist,'Enable','off');
set(handles.fuen_int_3_dist,'Enable','off');
    end
elseif get(handles.fuen_comp_dist,'Value')==1
    axes(handles.formula_2)
    formula = imread('imagenes\term_fuente_dist.png');
    image(formula);
    axis off
    axis image
    set(handles.bip_int_1_dist,'Enable','off');
set(handles.bip_int_2_dist,'Enable','off');
set(handles.bip_int_3_dist,'Enable','off');
    set(handles.nu_1_dist,'Enable','off');
set(handles.nu_2_dist,'Enable','off');
set(handles.nu_3_dist,'Enable','off');
set(handles.nu_4_dist,'Enable','off');
    set(handles.beta_1_dist,'Enable','on');
set(handles.beta_2_dist,'Enable','on');
set(handles.beta_3_dist,'Enable','on');
    set(handles.fuen_int_1_dist,'Enable','on');
set(handles.fuen_int_2_dist,'Enable','on');
set(handles.fuen_int_3_dist,'Enable','on');
else
    axes(handles.formula_2)
    formula = imread('imagenes\distancia.png');
    image(formula);
    axis off
    axis image
    set(handles.bip_int_1_dist,'Enable','off');
set(handles.bip_int_2_dist,'Enable','off');
set(handles.bip_int_3_dist,'Enable','off');
    set(handles.nu_1_dist,'Enable','off');
set(handles.nu_2_dist,'Enable','off');
set(handles.nu_3_dist,'Enable','off');
set(handles.nu_4_dist,'Enable','off');
    set(handles.beta_1_dist,'Enable','off');
set(handles.beta_2_dist,'Enable','off');
set(handles.beta_3_dist,'Enable','off');
    set(handles.fuen_int_1_dist,'Enable','off');
set(handles.fuen_int_2_dist,'Enable','off');
set(handles.fuen_int_3_dist,'Enable','off');
end


function fuen_int_1_dist_Callback(hObject, eventdata, handles)
```

```
function fuen_int_1_dist_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bip_int_1_dist_Callback(hObject, eventdata, handles)

function bip_int_1_dist_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function nu_1_dist_Callback(hObject, eventdata, handles)

function nu_1_dist_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function beta_1_dist_Callback(hObject, eventdata, handles)

function beta_1_dist_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function x_param_1_Callback(hObject, eventdata, handles)

function x_param_1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function y_param_1_Callback(hObject, eventdata, handles)

function y_param_1_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function z_param_1_Callback(hObject, eventdata, handles)

function z_param_1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function x_centro_1_Callback(hObject, eventdata, handles)

function x_centro_1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function y_centro_1_Callback(hObject, eventdata, handles)

function y_centro_1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function z_centro_1_Callback(hObject, eventdata, handles)

function z_centro_1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function x_cart_2_Callback(hObject, eventdata, handles)

function x_cart_2_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function y_cart_1_Callback(hObject, eventdata, handles)

function y_cart_1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function z_cart_1_Callback(hObject, eventdata, handles)

function z_cart_1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function menu_front_Callback(hObject, eventdata, handles)
if get(handles.menu_front,'Value')==1
    set(handles.front_cart,'Visible','off')
    set(handles.front_param,'Visible','on')
    seleccion_dimensiones_dist_Callback(hObject, eventdata, handles)
elseif get(handles.menu_front,'Value')==2
    set(handles.front_param,'Visible','off')
    set(handles.front_cart,'Visible','on')
    seleccion_dimensiones_dist_Callback(hObject, eventdata, handles)
end

function menu_front_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function y_cart_2_Callback(hObject, eventdata, handles)

function y_cart_2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
```

```
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function z_cart_2_Callback(hObject, eventdata, handles)

function z_cart_2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function x_cart_1_Callback(hObject, eventdata, handles)

function x_cart_1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function repetir_ClickedCallback(hObject, eventdata, handles)
global x y d d1 d2 X Y Z p p1 p2 tipo_calculo

if tipo_calculo==1
    if get(handles.animacion,'Value')==0
        if get(handles.euler,'Value')==1&&get(handles.rk4,'Value')==1
            set(handles.panel_resultados,'Visible','on')
            subplot(1,1,1); plot(0,0)
            comparacionh(x,y,'Euler','Runge-Kutta 4')
        else
        set(handles.panel_resultados,'Visible','on')
        subplot(1,1,1); plot(0,0)
        dibujar(x)
        end
    else
        if get(handles.euler,'Value')==1&&get(handles.rk4,'Value')==1
            set(handles.panel_resultados,'Visible','on')
            subplot(1,1,1); plot(0,0)
            animacionh(x,y,'Euler','Runge-Kutta 4',p)
        else
            set(handles.panel_resultados,'Visible','on')
            subplot(1,1,1); plot(0,0)
            animacion(x,p)
        end
```

```
    end

elseif tipo_calculo==2
    if get(handles.animacion_dist,'Value')==0
        if get(handles.euler_dist,'Value')==1&&get
(handles.rk4_dist,'Value')==1
            set(handles.panel_resultados,'Visible','on')
            subplot(1,1,1); plot(0,0)
            comparacionh_dist(x,y,d1,d2,'Euler','Runge-Kutta 4')
        else
        set(handles.panel_resultados,'Visible','on')
        subplot(1,1,1); plot(0,0)
        dibujar_dist(x,d)
        end
    else
        if get(handles.euler_dist,'Value')==1&&get
(handles.rk4_dist,'Value')==1
            set(handles.panel_resultados,'Visible','on')
            subplot(1,1,1); plot(0,0)
            comparacionh_dist(x,y,d1,d2,'Euler','Runge-Kutta 4')

        elseif get(handles.tipo_animacion_1,'Value')==1
            set(handles.panel_resultados,'Visible','on')
            subplot(1,1,1); plot(0,0)
            animacion1(x,d,X,Y,Z,p1,p2)
        elseif get(handles.tipo_animacion_1,'Value')==2
            set(handles.panel_resultados,'Visible','on')
            subplot(1,1,1); plot(0,0)
            animacion2(x,d,X,Y,Z,p1,p2)
        elseif get(handles.tipo_animacion_1,'Value')==3
            set(handles.panel_resultados,'Visible','on')
            subplot(1,1,1); plot(0,0)
            animacion3(x,d,X,Y,Z,p1,p2)
        end
    end
end



function i_inf_Callback(hObject, eventdata, handles)

function i_inf_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end

function i_sup_Callback(hObject, eventdata, handles)

function i_sup_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function j_inf_Callback(hObject, eventdata, handles)

function j_inf_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function j_sup_Callback(hObject, eventdata, handles)

function j_sup_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function tam_Callback(hObject, eventdata, handles)

function tam_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function tam_param_Callback(hObject, eventdata, handles)

function tam_param_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function exportar_ClickedCallback(hObject, eventdata, handles)
global T x y d d1 d2 X Y Z tipo_calculo

[n,~]=size(x); T=T(1:n)';
if tipo_calculo==1
    if (get(handles.euler,'Value')==1&&get(handles.rk4,'Value')==0)||...
        (get(handles.rk4,'Value')==1&&get(handles.euler,'Value')==0)
        res=[x T];
        [nombre,ruta]=uiputfile('*.txt','Exportar resultados');
        direccion=strcat(ruta,nombre);
        dlmwrite(direccion,res,'delimiter','\t','precision','%.6f','newline','pc');

    elseif get(handles.euler,'Value')==1&&get(handles.rk4,'Value')==1
    res=[x y T];
    [nombre,ruta]=uiputfile('*.txt','Exportar resultados');
    direccion=strcat(ruta,nombre);
    dlmwrite(direccion,res,'delimiter','\t','precision','%.6f','newline','pc');
    end
elseif tipo_calculo==2;
    if ((get(handles.euler_dist,'Value')==1&&get(handles.rk4_dist,'Value')==0)||...
       (get(handles.rk4_dist,'Value')==1&&get(handles.euler_dist,'Value')==0))&&...
       get(handles.animacion_dist,'Value')==0
    d=[d;d(n-1)];
    res=[x d T];
    [nombre,ruta]=uiputfile('*.txt','Exportar resultados');
    direccion=strcat(ruta,nombre);
    dlmwrite(direccion,res,'delimiter','\t','precision','%.6f','newline','pc');

    elseif ((get(handles.euler_dist,'Value')==1&&get(handles.rk4_dist,'Value')==0)||...
       (get(handles.rk4_dist,'Value')==1&&get(handles.euler_dist,'Value')==0))&&...
       get(handles.animacion_dist,'Value')==1
    d=[d;d(n-1)];
    sep=zeros(n,1);
    for i=1:n
        sep(i,1)=NaN;
    end
    [m,~]=size(X);
    if m~=n
        for i=n+1:m
            sep(i,1)=1;
            x(i,:)=0;
            d(i)=0;
            T(i)=0;
        end
```

```
    end
    res=[x d T sep X Y Z];
    [nombre,ruta]=uiputfile('*.txt','Exportar resultados');
    direccion=strcat(ruta,nombre);
    dlmwrite(direccion,res,'delimiter','\t','precision','%.6f','newline','pc');

    elseif get(handles.euler_dist,'Value')==1&&get(handles.rk4_dist,'Value')==1
    d1=[d1;d1(n-1)]; d2=[d2;d2(n-1)];
    res=[x y d1 d2 T];
    [nombre,ruta]=uiputfile('*.txt','Exportar resultados');
    direccion=strcat(ruta,nombre);
    dlmwrite(direccion,res,'delimiter','\t','precision','%.6f','newline','pc');
    end
end

function cargar_datos_ClickedCallback(hObject, eventdata, handles)

[nombre,ruta]=uigetfile('*.txt','Cargar datos para representar');
direccion=strcat(ruta,nombre);
res=dlmread(direccion,'\t');
[tipo_problema,ok] = listdlg('ListString',{'Caso general sin animacion ->
1 metodo';...
    'Caso general sin animacion -> Comparacion 2 metodos';...
    'Caso general con animacion -> 1 metodo';...
    'Caso general con animacion -> Comparacion 2 metodos';...
    'Funcion distancia sin animacion -> 1 metodo';...
    'Funcion distancia sin animacion -> Comparacion 2 metodos';...
    'Funcion distancia con animacion 1 -> 1 metodo';...
    'Funcion distancia con animacion 2 -> 1 metodo';...
    'Funcion distancia con animacion 3 -> 1 metodo'},...
    'SelectionMode','single','Name','Tipo de problema','PromptString',...
    'Seleccione tipo de problema','ListSize',[300 150]);
if ok==1
    switch tipo_problema
        case 1
            fig_resultados= axes('Parent',handles.panel_resultados,
'Position',[.1 .1 .9 .9]);
            [~,n]=size(res);
            x=res(:,1:n-1);
            subplot(1,1,1); plot(0,0)
            dibujar(x)
        case 2
            fig_resultados= axes('Parent',handles.panel_resultados,
'Position',[.1 .1 .9 .9]);
            [~,n]=size(res);
            x=res(:,1:(n-1)/2);
```

```
            y=res(:,(n-1)/2+1:n-1);
            subplot(1,1,1); plot(0,0)
            comparacionh(x,y,'Euler','Runge-Kutta 4')
        case 3
            fig_resultados= axes('Parent',handles.panel_resultados,
'Position',[.1 .1 .9 .9]);
            [~,n]=size(res);
            x=res(:,1:n-1);
            subplot(1,1,1); plot(0,0)
            animacion(x,0.01)
        case 4
            fig_resultados= axes('Parent',handles.panel_resultados,
'Position',[.1 .1 .9 .9]);
            [~,n]=size(res);
            x=res(:,1:(n-1)/2);
            y=res(:,(n-1)/2+1:n-1);
            subplot(1,1,1); plot(0,0)
            animacionh(x,y,'Euler','Runge-Kutta 4',0.01)
        case 5
            fig_resultados= axes('Parent',handles.panel_resultados,
'Position',[.1 .1 .9 .9]);
            [~,n]=size(res);
            x=res(:,1:n-2);
            d=res(:,n-1);
            subplot(1,1,1); plot(0,0)
            dibujar_dist(x,d)
        case 6
            fig_resultados= axes('Parent',handles.panel_resultados,
'Position',[.1 .1 .9 .9]);
            [~,n]=size(res);
            x=res(:,1:(n-3)/2);
            y=res(:,(n-3)/2+1:n-3);
            d1=res(:,n-2);
            d2=res(:,n-1);
            subplot(1,1,1); plot(0,0)
            comparacionh_dist(x,y,d1,d2,'Euler','Runge-Kutta 4')
        case 7
            fig_resultados= axes('Parent',handles.panel_resultados,
'Position',[.1 .1 .9 .9]);
            a=0;
            [~,n]=size(res);
            q=isnan(res(1,:));
            for i=1:n
                if q(i)==1
                    a=i;
                end
```

```
            end
            c=find(res(:,a)>0,1);
            x=res(1:c-1,1:a-3);
            d=res(1:c-1,a-2);
            b=(n-a)/3;
            X=res(:,a+1:a+b);
            Y=res(:,a+b+1:a+2*b);
            Z=res(:,a+2*b+1:n);
            subplot(1,1,1); plot(0,0)
            animacion1(x,d,X,Y,Z,0.01,0.01)
        case 8
            fig_resultados= axes('Parent',handles.panel_resultados,
'Position',[.1 .1 .9 .9]);
            a=0;
            [~,n]=size(res);
            q=isnan(res(1,:));
            for i=1:n
                if q(i)==1
                    a=i;
                end
            end
            c=find(res(:,a)>0,1);
            x=res(1:c-1,1:a-3);
            d=res(1:c-1,a-2);
            b=(n-a)/3;
            X=res(:,a+1:a+b);
            Y=res(:,a+b+1:a+2*b);
            Z=res(:,a+2*b+1:n);
            subplot(1,1,1); plot(0,0)
            animacion2(x,d,X,Y,Z,0.01,0.01)
        case 9
            fig_resultados= axes('Parent',handles.panel_resultados,
'Position',[.1 .1 .9 .9]);
            a=0;
            [~,n]=size(res);
            q=isnan(res(1,:));
            for i=1:n
                if q(i)==1
                    a=i;
                end
            end
            c=find(res(:,a)>0,1);
            x=res(1:c-1,1:a-3);
            d=res(1:c-1,a-2);
            b=(n-a)/3;
            X=res(:,a+1:a+b);
```

```
            Y=res(:,a+b+1:a+2*b);
            Z=res(:,a+2*b+1:n);
            subplot(1,1,1); plot(0,0)
            animacion3(x,d,X,Y,Z,0.01,0.01)
    end
end
```

6.2.2 fun_prueba.m

```
function f=fun_prueba(t,x)
global texto_fun_prueba

f=feval(texto_fun_prueba,t,x);
```

6.2.3 fun_prueba_2.m

```
function f=fun_prueba_2(t,x)
global texto_fun_prueba_2

f=feval(texto_fun_prueba_2,t,x);
```

6.2.4 beta.m

```
function f=beta(t,x)
global texto_beta
f=feval(texto_beta,t,x);
```

6.2.5 term_fuente.m

```
function f=term_fuente(t,x)
global texto_term_fuente

f=feval(texto_term_fuente,t,x);
```

6.2.6 funcionobjetivo.m

```
function f=funcionobjetivo(y)
global cx cy cz dim lambda1

if dim==1
   f=sqrt((y(2)-y(1))^2)+1/(2*lambda1)*((y(1)-cx)^2);

elseif dim==2
    f=sqrt((y(3)-y(1))^2+(y(4)-y(2))^2)+...
        1/(2*lambda1)*((y(3)-cx)^2+(y(4)-cy)^2);
```

```
elseif dim==3
    f=sqrt((y(4)-y(1))^2+(y(5)-y(2))^2+(y(6)-y(3))^2)+...
        1/(2*lambda1)*((y(4)-cx)^2+(y(5)-cy)^2+(y(6)-cz)^2);

end
```

6.2.7 funcionrestricciones.m

```
function f=funcionrestricciones(x)
global t texto_funcionrestricciones

f=feval(texto_funcionrestricciones,t,x);
```

6.2.8 funciondistancia.m

```
function f=funciondistancia(y)
global dim sol

if dim==1
   f=sqrt((sol(1)-y(1))^2);

elseif dim==2
    f=sqrt((sol(1)-y(1))^2+(sol(2)-y(2))^2);

elseif dim==3
    f=sqrt((sol(1)-y(1))^2+(sol(2)-y(2))^2+(sol(3)-y(3))^2);

end
```

6.2.9 centro.m

```
function [x,y,z]=centro(t)
global x_centro y_centro z_centro

x=feval(x_centro,t);
y=feval(y_centro,t);
z=feval(z_centro,t);
```

6.2.10 cartesianas.m

```
function [x,y,z]=cartesianas(t,tam)
global dim x_cart_1 x_cart_2 y_cart_1 y_cart_2 z_cart_1 z_cart_2

a=str2func(strcat('@(t)',x_cart_1));
x_lim_izq=feval(a,t);
b=str2func(strcat('@(t)',x_cart_2));
```

```
x_lim_der=feval(b,t);
incr=(x_lim_der-x_lim_izq)/tam;
x=x_lim_izq:incr:x_lim_der;
y=0; z=0;
if dim==2
    x=[x x];
    y=zeros(1,2*tam+2);
    for l=1:tam+1
        c=str2func(strcat('@(t,x)',y_cart_1));
        y(l)=feval(c,t,x(l));
    end
    for l=tam+2:2*tam+2
        d=str2func(strcat('@(t,x)',y_cart_2));
        y(l)=feval(d,t,x(l));
    end

elseif dim==3
    X=zeros(tam+1,tam+1);
    Y=zeros(tam+1,tam+1);
    Z1=zeros(tam+1,tam+1);
    Z2=zeros(tam+1,tam+1);

    c=str2func(strcat('@(t,x)',y_cart_1));
    d=str2func(strcat('@(t,x)',y_cart_2));
    e=str2func(strcat('@(t,x,y)',z_cart_1));
    f=str2func(strcat('@(t,x,y)',z_cart_2));

    for i=1:tam+1
        X(i,:)=x(i);
    end
    for j=1:tam+1
        Y(1,j)=feval(c,t,X(1,j));Y(tam+1,j)=feval(c,t,X(tam+1,j));
        Z1(1,j)=feval(e,t,X(1,j),Y(1,j));Z1(tam+1,j)=feval(e,t,
X(tam+1,j),Y(tam+1,j));
        Z2(1,j)=feval(e,t,X(1,j),Y(1,j));Z2(tam+1,j)=feval(e,t,
X(tam+1,j),Y(tam+1,j));
    end

    for i=2:tam
        y_lim_inf=feval(c,t,x(i));
        y_lim_sup=feval(d,t,x(i));
        incr_y=(y_lim_sup-y_lim_inf)/tam;
        malla_y=y_lim_inf:incr_y:y_lim_sup;
        Y(i,:)=malla_y;
        for j=1:tam+1
            Z1(i,j)=feval(e,t,X(i,j),Y(i,j));
```

```
            Z2(i,j)=feval(f,t,X(i,j),Y(i,j));
        end

    end
    X=real(X);Y=real(Y);Z1=real(Z1);Z2=real(Z2);
    Z=[Z1;Z2];
    x=[X;X];
    y=[Y;Y];
    z=Z;

end
```

6.2.11 parametricas.m

```
function [x,y,z]=parametricas(t,i,j)
global x_param y_param z_param

a=str2func(strcat('@(t,i,j)',x_param));
b=str2func(strcat('@(t,i,j)',y_param));
c=str2func(strcat('@(t,i,j)',z_param));
x=feval(a,t,i,j);
y=feval(b,t,i,j);
z=feval(c,t,i,j);
```

6.2.12 dibujar.m

```
function dibujar(x)
global dim T
pos=zeros(1,dim);
leyenda={'valores de x';'valores de y';'valores de z'};
if dim>1
    for i=1:dim
        pos(1,i)=2*i;
        subplot(dim,2,2*(i-1)+1); plot(T,x(:,i))
        xlabel('valores de t'); ylabel(leyenda(i));
    end
    if dim==2
        subplot(dim,2,pos);plot(x(:,1),x(:,2))
        xlabel(leyenda(1)); ylabel(leyenda(2));
    elseif dim==3
        subplot(dim,2,pos);xlabel(leyenda(1)); ylabel(leyenda(2));
zlabel(leyenda(3));
        set(gca,'CameraPosition',[10,10,10]);
        plot3(x(:,1),x(:,2),x(:,3))
        grid on
    end
```

```
else
    plot(T,x(:,1))
    xlabel('valores de t'); ylabel(leyenda(1));
end
```

6.2.13 dibujar_dist.m

```
function dibujar_dist(x,d)
global dim T m
pos=zeros(1,dim);
leyenda={'valores de x';'valores de y';'valores de z'};
if dim>1
    for i=1:dim
        pos(1,i)=2*i;
        subplot(dim+1,2,2*(i-1)+1); plot(T,x(:,i))
        xlabel('valores de t'); ylabel(leyenda(i));
    end
    subplot(dim+1,2,2*dim+1); plot(T(1:m),d)
    xlabel('valores de t'); ylabel('distancia');
    if dim==2
        subplot(dim+1,2,pos);plot(x(:,1),x(:,2))
        xlabel(leyenda(1)); ylabel(leyenda(2));
    elseif dim==3
        subplot(dim+1,2,pos);plot3(x(:,1),x(:,2),x(:,3))
        xlabel(leyenda(1)); ylabel(leyenda(2));zlabel(leyenda(3))
        grid on
    end
else
    subplot(1,2,1); plot(T,x(:,1))
    xlabel('valores de t'); ylabel(leyenda(1));
    subplot(1,2,2); plot(T(1:m),d)
    xlabel('valores de t'); ylabel('distancia');
end
```

6.2.14 comparacionh.m

```
function comparacionh(x,y,metodo1,metodo2)
global dim T
pos=zeros(1,dim);
leyenda={'Valores de x';'Valores de y';'Valores de z'};
if dim>1
    for i=1:dim
        pos(1,i)=4*(i-1)+2;
        subplot(dim,4,4*(i-1)+1); plot(T,x(:,i))
        xlabel('Valores de t'); ylabel(leyenda(i));title(metodo1);
        subplot(dim,4,4*(i-1)+3); plot(T,y(:,i))
```

```
        xlabel('Valores de t'); ylabel(leyenda(i));title(metodo2);
    end
    if dim==2
        subplot(dim,4,pos);plot(x(:,1),x(:,2))
        xlabel(leyenda(1)); ylabel(leyenda(2));title(metodo1);
        subplot(dim,4,pos+2);plot(y(:,1),y(:,2))
        xlabel(leyenda(1)); ylabel(leyenda(2));title(metodo2);

    else if dim==3
        subplot(dim,4,pos);
        xlabel(leyenda(1)); ylabel(leyenda(2));zlabel(leyenda(3));
title(metodo1);
        set(gca,'CameraPosition',[10,10,10]);plot3(x(:,1),x(:,2),
x(:,3))
        grid on
        subplot(dim,4,pos+2);
        xlabel(leyenda(1)); ylabel(leyenda(2));zlabel(leyenda(3));
title(metodo2);
        set(gca,'CameraPosition',[10,10,10]);plot3(y(:,1),y(:,2),
y(:,3))
        grid on
        end
    end
else
    subplot(1,2,1);plot(T,x(:,1));
    xlabel('valores de t'); ylabel(leyenda(1));title(metodo1);
    subplot(1,2,2);plot(T,y(:,1));
    xlabel('valores de t'); ylabel(leyenda(1));title(metodo2);
end
```

6.2.15 comparacionh_dist.m

```
function comparacionh_dist(x,y,d1,d2,metodo1,metodo2)
global dim T m
pos=zeros(1,dim);
leyenda={'Valores de x';'Valores de y';'Valores de z'};
if dim>1
    for i=1:dim
        pos(1,i)=4*(i-1)+2;
        subplot(dim+1,4,4*(i-1)+1); plot(T,x(:,i))
        xlabel('Valores de t'); ylabel(leyenda(i));title(metodo1);
        subplot(dim+1,4,4*(i-1)+3); plot(T,y(:,i))
        xlabel('Valores de t'); ylabel(leyenda(i));title(metodo2);
    end
    subplot(dim+1,4,4*dim+1); plot(T(1:m),d1)
    xlabel('valores de t'); ylabel('distancia');title(metodo1);
```

```
    subplot(dim+1,4,4*dim+3); plot(T(1:m),d2)
    xlabel('valores de t'); ylabel('distancia');title(metodo2);
    if dim==2
        subplot(dim+1,4,pos);plot(x(:,1),x(:,2))
        xlabel(leyenda(1)); ylabel(leyenda(2));title(metodo1);
        subplot(dim+1,4,pos+2);plot(y(:,1),y(:,2))
        xlabel(leyenda(1)); ylabel(leyenda(2));title(metodo2);

    else if dim==3
        subplot(dim+1,4,pos);plot3(x(:,1),x(:,2),x(:,3))
        xlabel(leyenda(1)); ylabel(leyenda(2));zlabel(leyenda(3));
title(metodo1);
        grid on
        subplot(dim+1,4,pos+2);plot3(y(:,1),y(:,2),y(:,3))
        xlabel(leyenda(1)); ylabel(leyenda(2));zlabel(leyenda(3));
title(metodo2);
        grid on
        end
    end
else
    subplot(2,2,1);plot(T,x(:,1));
    xlabel('valores de t'); ylabel(leyenda(1));title(metodo1);
    subplot(2,2,2);plot(T,y(:,1));
    xlabel('valores de t'); ylabel(leyenda(1));title(metodo2);
    subplot(2,2,3); plot(T(1:m),d1)
    xlabel('valores de t'); ylabel('distancia');title(metodo1);
    subplot(2,2,4); plot(T(1:m),d2)
    xlabel('valores de t'); ylabel('distancia');title(metodo2);
end
```

6.2.16 animacion.m

```
function animacion(x,p)
global dim T m
pos=zeros(1,dim);
leyenda={'valores de x';'valores de y';'valores de z'};
if dim>1
    for i=1:dim
        pos(1,i)=2*i;
        subplot(dim,2,2*(i-1)+1);
        xlabel('valores de t'); ylabel(leyenda(i));
        for j=1:m+1
            hold on
            plot(T(j),x(j,i))
            pause (p)
        end
```

```
    end
    if dim==2
        subplot(dim,2,pos);
        xlabel(leyenda(1)); ylabel(leyenda(2));
        for j=1:m+1
            hold on
            plot(x(j,1),x(j,2))
            pause (p)
        end
    elseif dim==3
        subplot(dim,2,pos);
        xlabel(leyenda(1)); ylabel(leyenda(2));zlabel(leyenda(3));
        set(gca,'CameraPosition',[10,10,10]);
        grid on
        for j=1:m+1
            hold on
            plot3(x(j,1),x(j,2),x(j,3))
            pause (p)
        end
    end
else
    plot(T,x(:,1))
    xlabel('valores de t'); ylabel(leyenda(1));
end
```

6.2.17 animacionh.m

```
function animacionh(x,y,metodo1,metodo2,p)
global dim T m
pos=zeros(1,dim);
leyenda={'Valores de x';'Valores de y';'Valores de z'};


if dim>1
    for i=1:dim
        pos(1,i)=4*(i-1)+2;
        for j=1:m+1
            subplot(dim,4,4*(i-1)+1);
            plot(T(j),x(j,i))
            xlabel('Valores de t'); ylabel(leyenda(i));
title(metodo1);
            hold on
            subplot(dim,4,4*(i-1)+3);
            plot(T(j),y(j,i))
            xlabel('Valores de t'); ylabel(leyenda(i));
title(metodo2);
```

```
            hold on
            pause (p)
        end
    end
    if dim==2
        for j=1:m+1
            subplot(dim,4,pos);
            plot(x(j,1),x(j,2),'d')
            xlabel(leyenda(1)); ylabel(leyenda(2));title(metodo1);
            hold on
            subplot(dim,4,pos+2);
            plot(y(j,1),y(j,2),'d')
            xlabel(leyenda(1)); ylabel(leyenda(2));title(metodo2);
            hold on
            pause (p)
        end

    elseif dim==3
        for j=1:m+1
            subplot(dim,4,pos);
            plot3(x(j,1),x(j,2),x(j,3),'d')
            xlabel(leyenda(1)); ylabel(leyenda(2));
zlabel(leyenda(3));title(metodo1);
            grid on
            hold on
            subplot(dim,4,pos+2);
            plot3(y(j,1),y(j,2),y(j,3),'d')
            xlabel(leyenda(1)); ylabel(leyenda(2));
zlabel(leyenda(3));title(metodo2);
            grid on
            hold on
            pause (p)
        end
    end
else
    subplot(1,2,1);plot(T,x(:,1));
    xlabel('valores de t'); ylabel(leyenda(1));title(metodo1);
    subplot(1,2,2);plot(T,y(:,1));
    xlabel('valores de t'); ylabel(leyenda(1));title(metodo2);
end
```

6.2.18 animacion1.m

```
function animacion1(x,d,X,Y,Z,p1,p2)
global dim T m tipo_front
```

```
[~,n]=size(X);
pos=zeros(1,dim);
leyenda={'valores de x';'valores de y';'valores de z'};


if dim==2
    for i=1:dim
        pos(1,i)=2*i;
        subplot(dim+1,2,2*(i-1)+1);
        xlabel('valores de t'); ylabel(leyenda(i));
        for j=1:m+1
            hold on
            plot(T(j),x(j,i))
            pause (p1)
        end
    end
    subplot(dim+1,2,2*dim+1);
    xlabel(leyenda(1)); ylabel(leyenda(2));
        for j=1:m+1
            hold on
            plot(x(j,1),x(j,2))
            pause (p1)
        end

    subplot(dim+1,2,2*dim+2);
    xlabel('valores de t');ylabel('distancia');
    for j=1:m
        hold on
        plot(T(j),d(j))
        pause (p1)
    end

    subplot(dim+1,2,pos);
    xlabel(leyenda(1)); ylabel(leyenda(2));
    if tipo_front==1
        for j=1:m+1
            plot(x(j,1),x(j,2),'d',X(j,:),Y(j,:),'r')
            pause (p2)
        end
    elseif tipo_front==2
        for j=1:m+1
            plot(x(j,1),x(j,2),'d',X(j,1:n/2),Y(j,1:n/2),'r',
X(j,n/2+1:n),Y(j,n/2+1:n),'r')
            pause (p2)
        end
    end
```

```
elseif dim==3

    for i=1:dim
        pos(1,i)=2*i;
        subplot(dim,2,2*(i-1)+1);
        xlabel('valores de t'); ylabel(leyenda(i));
        for j=1:m+1
            hold on
            plot(T(j),x(j,i))
            pause (p1)
        end
    end

    subplot(dim,2,2*dim);
    xlabel('valores de t');ylabel('distancia');
    for j=1:m
        hold on
        plot(T(j),d(j))
        pause (p1)
    end

    subplot(dim+1,2,pos);
    xlabel(leyenda(1)); ylabel(leyenda(2));zlabel(leyenda(3))
    set(gca,'CameraPosition',[10,10,10]);
for i=1:dim
       subplot(dim,2,2*(i-1)+1);
       plot(T,x(:,i),'')
    end
    subplot(dim,2,2*dim);
    plot(T(1:m),d,'')
    subplot(dim+1,2,pos);
    if tipo_front==2
        for j=1:m+1
            plot3(x(j,1),x(j,2),x(j,3),'d')
            hold on
            grid on
            surf(X(2*n*(j-1)+1:2*n*j,:),Y(2*n*(j-1)+1:2*n*j,:),...
                Z(2*n*(j-1)+1:2*n*j,:),'FaceColor','red',
'FaceAlpha',0.6)
            hold off
            pause (p2)
        end
    elseif tipo_front==1
        for j=1:m
            plot3(x(j,1),x(j,2),x(j,3),'d')
```

```
            hold on
            grid on
            surf(X(n*(j-1)+1:n*j,:),Y(n*(j-1)+1:n*j,:),...
                Z(n*(j-1)+1:n*j,:),'FaceColor','red',
'FaceAlpha',0.6)
            hold off
            pause (p2)
        end
    end
else
plot(T,x(:,1))
xlabel('valores de t'); ylabel(leyenda(1));
end
```

6.2.19 animacion2.m

```
function animacion2(x,d,X,Y,Z,p1,p2)
global dim T m tipo_front

[~,n]=size(X);
pos=zeros(1,dim);
leyenda={'valores de x';'valores de y';'valores de z'};
H=zeros(m+1,n);

for l=1:m+1
    H(l,:)=T(l);
end


if dim==2
    for i=1:dim
        pos(1,i)=2*i;
        subplot(dim+1,2,2*(i-1)+1);
        xlabel('valores de t'); ylabel(leyenda(i));
        for j=1:m+1
            hold on
            plot(T(j),x(j,i))
            pause (p1)
        end
    end
    subplot(dim+1,2,2*dim+1);
    xlabel(leyenda(1)); ylabel(leyenda(2));
    for j=1:m+1
        hold on
        plot(x(j,1),x(j,2))
        pause (p1)
```

```
    end

    subplot(dim+1,2,2*dim+2);
    xlabel('valores de t');ylabel('distancia');
    for j=1:m
        hold on
        plot(T(j),d(j))
        pause (p1)
    end

    subplot(dim+1,2,pos);
    xlabel(leyenda(1)); ylabel(leyenda(2));
    set(gca,'CameraPosition',[10,10,10]);
    if tipo_front==1
        for j=1:m+1
            hold on
            grid on
            plot3(x(j,1),x(j,2),T(j),'d',X(j,:),Y(j,:),
H(j,:),'r')
            pause (p2)
        end
    elseif tipo_front==2
        for j=1:m+1
            hold on
            grid on
            plot3(x(j,1),x(j,2),T(j),'d',X(j,1:n/2),Y(j,1:n/2),
H(j,1:n/2),'r',X(j,n/2+1:n),Y(j,n/2+1:n),H(j,n/2+1:n),
'r')
            pause (p2)
        end
    end

elseif dim==3

    for i=1:dim
        pos(1,i)=2*i;
        subplot(dim,2,2*(i-1)+1);
        xlabel('valores de t'); ylabel(leyenda(i));
        for j=1:m+1
            hold on
            plot(T(j),x(j,i))
            pause (p1)
        end
    end

    subplot(dim,2,2*dim);
```

```
    xlabel('valores de t');ylabel('distancia');
    for j=1:m
        hold on
        plot(T(j),d(j))
        pause (p1)
    end

    subplot(dim+1,2,pos);
    xlabel(leyenda(1)); ylabel(leyenda(2));zlabel(leyenda(3));
    set(gca,'CameraPosition',[10,10,10]);
    for j=1:m+1
        hold on
        plot3(x(j,1),x(j,2),x(j,3))
        grid on
        pause (p2)
    end

else
    plot(T,x(:,1))
    xlabel('valores de t'); ylabel(leyenda(1));
end
```

6.2.20 animacion3.m

```
function animacion3(x,d,X,Y,Z,p1,p2)
global dim T m tipo_front

[~,n]=size(X);
pos=zeros(1,dim);
leyenda={'valores de x';'valores de y';'valores de z'};


if dim==2
    for i=1:dim
        pos(1,i)=2*i;
        subplot(dim+1,2,2*(i-1)+1);
        xlabel('valores de t'); ylabel(leyenda(i));
        for j=1:m+1
            hold on
            plot(T(j),x(j,i))
            pause (p1)
        end
    end
    subplot(dim+1,2,2*dim+1);
    xlabel(leyenda(1)); ylabel(leyenda(2));
        for j=1:m+1
```

```
            hold on
            plot(x(j,1),x(j,2))
            pause (p1)
        end

    subplot(dim+1,2,2*dim+2);
    xlabel('valores de t');ylabel('distancia');
    for j=1:m
        hold on
        plot(T(j),d(j))
        pause (p1)
    end

    subplot(dim+1,2,pos);
    xlabel(leyenda(1)); ylabel(leyenda(2));
    if tipo_front==1
        hold on
        h=plot(X(1,:),Y(1,:),'r');
        plot(x(1,1),x(1,2),'d')
        pause (p2)
        for j=2:m+1
            delete(h)
            h=plot(X(j,:),Y(j,:),'r');
            plot(x(j,1),x(j,2),'d')
            pause (p2)
        end
    elseif tipo_front==2
        hold on
        h=plot(X(1,1:n/2),Y(1,1:n/2),'r',X(1,n/2+1:n),Y(1,n/2+1:n),
'r');
        plot(x(1,1),x(1,2),'d');
        pause (p2)
        for j=2:m+1
            delete(h)
            h=plot(X(j,1:n/2),Y(j,1:n/2),'r',X(j,n/2+1:n),
Y(j,n/2+1:n),'r');
            plot(x(j,1),x(j,2),'d');
            pause (p2)
        end
    end

elseif dim==3

    for i=1:dim
        pos(1,i)=2*i;
        subplot(dim,2,2*(i-1)+1);
```

```
        xlabel('valores de t'); ylabel(leyenda(i));
        for j=1:m+1
            hold on
            plot(T(j),x(j,i))
            pause (p1)
        end
    end

    subplot(dim,2,2*dim);
    xlabel('valores de t');ylabel('distancia');
    for j=1:m
        hold on
        plot(T(j),d(j))
        pause (p1)
    end

    subplot(dim+1,2,pos);
    xlabel(leyenda(1)); ylabel(leyenda(2));zlabel(leyenda(3))
    set(gca,'CameraPosition',[10,10,10]);
for i=1:dim
       subplot(dim,2,2*(i-1)+1);
       plot(T,x(:,i),'')
    end
    subplot(dim,2,2*dim);
    plot(T(1:m),d,'')
    subplot(dim+1,2,pos);
    if tipo_front==1
        hold on
        grid on
        h=surf(X(n*(1-1)+1:n*1,:),Y(n*(1-1)+1:n*1,:),...
                Z(n*(1-1)+1:n*1,:),'FaceColor','red',
'FaceAlpha',0.6);
        plot3(x(1,1),x(1,2),x(1,3),'d')
        pause (p2)
        for j=2:m+1
            delete(h)
            h=surf(X(n*(j-1)+1:n*j,:),Y(n*(j-1)+1:n*j,:),...
                Z(n*(j-1)+1:n*j,:),'FaceColor','red',
'FaceAlpha',0.6);
            plot3(x(j,1),x(j,2),x(j,3),'d')
            pause (p2)
        end
    elseif tipo_front==2
        hold on
        grid on
        h=surf(X(2*n*(1-1)+1:2*n*1,:),Y(2*n*(1-1)+1:2*n*1,:),...
```

```
                Z(2*n*(1-1)+1:2*n*1,:),'FaceColor','red',
'FaceAlpha',0.6);
        plot3(x(1,1),x(1,2),x(1,3),'d')
        pause (p2)
        for j=2:m+1
            delete(h)
            h=surf(X(2*n*(j-1)+1:2*n*j,:),Y(2*n*(j-1)+1:2*n*j,:),...
                Z(2*n*(j-1)+1:2*n*j,:),'FaceColor','red',
'FaceAlpha',0.6);
            plot3(x(j,1),x(j,2),x(j,3),'d')
            pause (p2)
        end
    end
else
plot(T,x(:,1))
xlabel('valores de t'); ylabel(leyenda(1));
end
```

Bibliografía

[1] H. Attouch y M.-O. Czarnecki, *Asymptotic behavior of coupled dynamical systems with multiscale aspects*, J. Differential Equations, **248** (2010), 1315–1344.

[2] J.-P. Aubin, L'analyse non linéaire et ses motivations économiques, Masson, 1984.

[3] J.-P. Aubin y A. Cellina, Differential Inclusions, Springer, 1984.

[4] H. Brézis, Opérateurs maximaux monotones et semi-groupes de contractions dans les espaces de Hilbert, North-Holland, 1973

[5] F.H. Clarke, Optimization and Nonsmooth Analysis, 1990 (reimpresión de la edición de Wiley, 1983).

[6] A. Dontchev y F. Lempio, *Difference methods for differential inclusions: a survey*, SIAM Review, **34** (1992), 263–294.

[7] S.D. Flåam, J.-B. Hiriart-Urruty y A. Jourani, *Feasibility in finite time*, J. Dynamical and Control Systems, **15** (2009), 537–555.

[8] D.J. Higham y N.J. Higham, MATLAB Guide, SIAM, 2005.

[9] J.-B. Hiriart-Urruty y C. Lemaréchal, Convex Analysis and Minimization Algorithms I, Springer, 1993.

[10] S. Hu y N.S. Papageorgiou, Handbook of Multivalued Analysis, Vol. II: Applications, Kluwer, 2000.

[11] A. Iserles, A First Course in the Numerical Analysis of Differential Equations, Cambridge University Press, 1996.

[12] A. Kuntsevich y F. Kappel, *SolvOpt*. The Solver for Local Nonlinear Optimization Problems, 1997.
`www.kfunigraz.ac.at/imawww/kuntsevich/solvopt`

[13] M.A. Lifante, Aproximación numérica de flujos de potenciales no regulares, Proyecto Fin de Carrera, ETSIT, UPCT, 2012.

[14] J.H. Mathew y K.D. Fink, Métodos numéricos con MATLAB, Prentice Hall, 2000.

[15] R.T. Rockafellar, Convex Analysis, Princeton University Press, 1997 (1970, 1a. edición).

[16] R.T. Rockafellar y R. J.-B. Wets, Variational Analysis, Springer, 1998.