
# Aproximación numérica de flujos de potenciales no regulares

Proyecto Fin de Carrera

**E.T.S. Ingeniería de Telecomunicación**

María Adela Lifante Amorós

Junio 2012

# Agradecimientos

A mi familia por el apoyo de todos estos años, en especial a mis padres. A las amistades forjadas entre laboratorios y clases de la UPCT que sin el apoyo de ellos todo esto no habría sido posible.

Comentar que la realización de este proyecto surge gracias a la concesión de una beca de formación ofertada por la UPCT, además de mi notable pasión por las matemáticas.

No me puedo olvidar de Alberto, el director del proyecto, que me dio la oportunidad de ser parte de éste y me apoyó y animó en cada momento.

Por último me gustaría dedicar todo mi trabajo realizado durante estos años a Tomás, mi abuelo y gran persona, que nos acaba de dejar y siempre quiso verme terminar la carrera.

# Indice

# Introducción

Las ecuaciones diferenciales de tipo potencial,

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

tienen interés dado que

- Permiten obtener trayectorias de descenso del potenciasl $\phi$.
- Describen sistemas dinámicos de interés.

Sin embargo, tanto desde el punto de vista teórico como de las aplicaciones, surgen problemas en los que el potencial $\phi$ no es una función regular (derivable). No obstante, cuando es convexa puede definirse el concepto de subgradiente de $\phi$ 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 solamente ocurre cuando el potencial se puede derivar, lo que obliga a definir la subdiferencial como el conjunto de los subgradientes y a plantear inclusiones diferenciales del tipo

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

El objeto de este proyecto es estudiar la aproximación numérica de ecuaciones diferenciales con términos multivaluados generados por subdiferenciales de funciones (potenciales) convexas. La idea básica 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 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 metodología 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, lo 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 de la 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.

Es también remarcable que puede probarse la convergencia de los esquemas obtenidos usando la discretización de Euler y que los experimentos numéricos realizados sugieren asimismo la convergencia de los esquemas de Runge-Kutta. Además su implementación en MATLAB resulta relativamente cómoda y satisfactoria, como hemos constatado a lo largo de múltiples simulaciones.

Esta memoria comienza desarrollando los fundamentos teóricos acerca del análisis convexo e inclusiones diferenciales asociadas a subdiferenciales. Hemos decidido incluir un resumen 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 Ingenieía.

Una vez comentados los conceptos teóricos para la comprensión de la memoria, el siguiente capítulo denominado Algoritmos se ha descrito la estructura de los algoritmos, utilizando los métodos numéricos de Euler y Runge-Kutta, y de los códigos programados en MATLAB. A continuación se describen diferentes escenarios según el tipo de problema que se nos pueda presentar. Se finaliza cada escenario del capítulo ejemplificando lo explicado mediante simulaciones numéricas.

En el capítulo de Experimentos Numéricos se estudian problemas más complejos donde no se conozca explícitamente la expresión de $\phi$. Con ello se realizan varios experimentos numéricos en cada uno de los escenarios ya descritos anteriormente que permiten observar la bondad de los métodos.

Una parte importante de la memoria la forma la parte asociada a las aplicaciones de nuestros métodos, donde se estudian casos de análisis de circuitos eléctricos no lineales y también sistemas mecánicos que poseen fricción dry o de Coulomb. En este capítulo se describe el modelo físico y matemático de cada una de las aplicaciones. Seguidamente se realiza una adaptación de dicho problema a nuestros algoritmos implementados, y con ello se obtienen resultados visibles y con aplicación práctica.

Por último se presentan los códigos programados en MATLAB en los Anexos finales de la memoria, donde en ellos se pueden consultar cada uno de los algoritmos comentados durante la memoria y algún otro más añadido.

# Capítulo 1

# Fundamentos teóricos

En este capítulo se recopilan las definiciones y las propiedades básicas que se manejarán a largo de la memoria. Los tres primeros apartados pueden englobarse bajo el epígrafe de *Análisis Convexo*, mientras que en 1.4 se consideran diversos casos de ecuaciones diferenciales multivaluadas o inclusiones diferenciales asociadas a subdiferenciales.

La referencia básica en el campo del Análisis Convexo es el libro de Rockafellar [16]. 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 [6] y [17].

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. Es obligado mencionar también el manual enciclopédico de Hu y Papageorgiou [10] y la obra pionera de Brézis [4], aunque sea un texto difícil por su enfoque abstracto.

Finalmente, cabe decir que en [10], [9], [16] y [17] 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.

En el caso de 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 $\mathbf{dl}_{\mathcal{H}}(K, Q) \leq \mathbf{dl}_{\mathcal{H}}(K, C) + \mathbf{dl}_{\mathcal{H}}(Q, C)$.

Finalmente, 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

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

### 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).

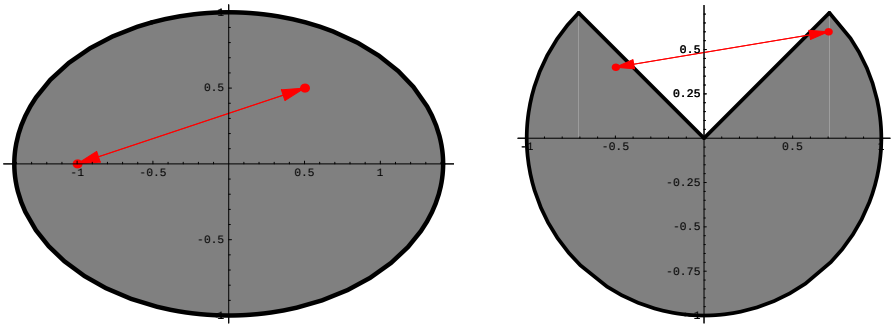
**Ejemplo 1.1** Veamos que la bola unidad $\mathcal{B}$ en $\mathbb{R}^N$ es un conjunto convexo. Tomemos para ello dos puntos $\mathbf{x}, \mathbf{y} \in \mathcal{B}$ y sea $0 \leq \lambda \leq 1$. Obviamente

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

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

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




Conjunto convexo Conjunto no convexo

Figura 1.1:

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, $A \in \mathcal{M}_{M \times N}(\mathbb{R})$ es una matriz y $\mathbf{z} \in \mathbb{R}^M$, se tiene que

  $$\mathbf{z} + AC = \{\mathbf{z} + A\mathbf{x} : \mathbf{x} \in C\} \subset \mathbb{R}^M$$

  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 $C$.

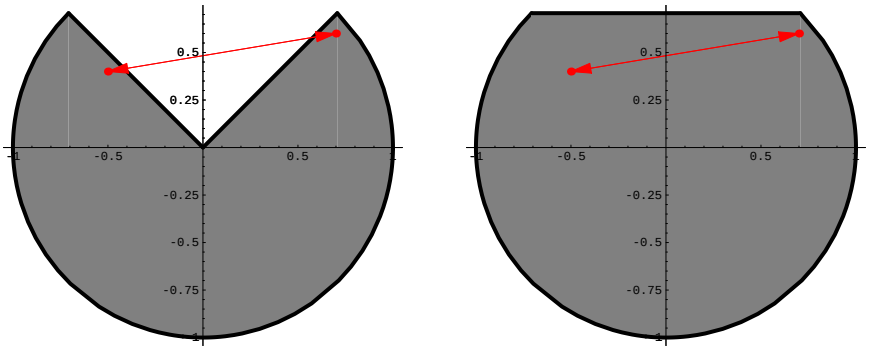
**Ejemplo 1.3** Si consideramos el conjunto de la derecha en la Figura 1.1, que puede describirse como

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

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

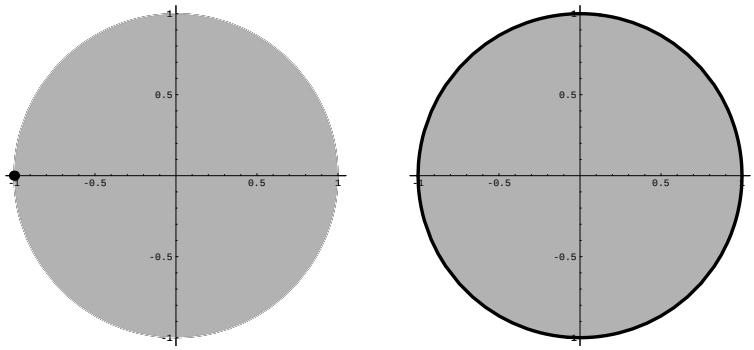
**Ejemplo 1.4** El conjunto $C = \left\{(x,y) \in \mathbb{R}^2 : x^2 + y^2 < 1\right\} \cup \{(-1,0)\}$ es convexo, luego $\mathrm{co}(C) = C$ (Figura 1.3). Sin embargo no es cerrado, por lo que $\overline{\mathrm{co}}(C) \neq \mathrm{co}(C)$. Claramente la envoltura convexa cerrada de $C$ es la bola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

$$\sum_{i=1}^{m} \lambda_i \mathbf{x}_i = \lambda_1\mathbf{x}_1 + (1-\lambda_1) \underbrace{\left( \sum_{i=2}^{m} \frac{\lambda_i}{(1-\lambda_1)} \mathbf{x}_i \right)}_{\mathbf{y}} \tag{1.9}$$




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

Figura 1.3: Envoltura convexa cerrada

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 $\mathrm{co}(C)$ 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

$$\mathrm{co}\,(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_p) = \left\{ \sum_{i=0}^{p} \lambda_i \mathbf{x}_i : 0 \leq \lambda_i,\ \sum_{i=0}^{p} \lambda_i = 1 \right\} \tag{1.10}$$

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.

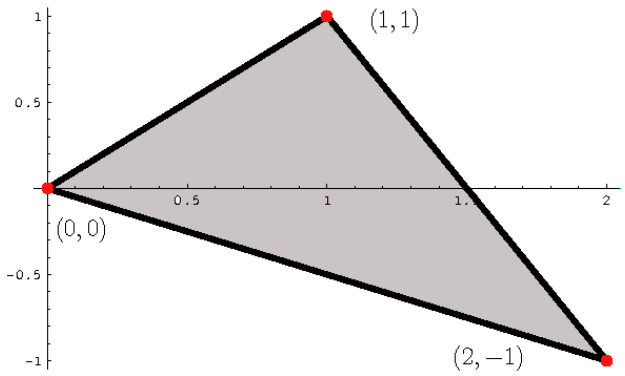



Figura 1.4: 2-simplex

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*

$$\mathrm{co}(C) = \left\{ \sum_{i=1}^{N+1} \lambda_i \mathbf{x}_i : 0 \leq \lambda_i, \sum_{i=1}^{N+1} \lambda_i = 1,\ \mathbf{x}_i \in C \right\} \tag{1.11}$$

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

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

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 $\mathbf{x} \in \overline{C}$, $\mathbf{z} \in \text{int}(C)$, se tiene que para cada $0 < \lambda \leq 1$,*

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

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

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

$$\langle\mathbf{x} - \hat{\mathbf{z}},\, \mathbf{z} - \hat{\mathbf{z}}\rangle \leq 0, \qquad \forall\, \mathbf{z} \in C. \tag{1.12}$$

De (1.12) 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 (1.12)

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

  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 por $\mathrm{proj}(\mathbf{x}; C)$ (del inglés *projection*).

**Proposición 1.2** *Sea $C \subset \mathbb{R}^N$ un conjunto convexo cerrado no vacío. Si $\mathbf{x} \notin C$, se tiene que* $\mathrm{proj}(\mathbf{x}; C) \in \partial C$[(1)]. *Además, el campo vectorial proyección ortogonal* $\mathrm{proj}(\cdot; C) : \mathbb{R}^N \longrightarrow C$ *es Lipschitz.*

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 $\mathbf{z} \in C$,*

$$\langle \mathbf{u}_0,\, \mathbf{z} \rangle \leq \langle \mathbf{u}_0,\, \mathbf{x}_0 \rangle - \varepsilon, \tag{1.13}$$

**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 (1.13), 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

$$\sigma_D(\mathbf{u}) = \sup_{\mathbf{x} \in D} \langle \mathbf{u}, \mathbf{x} \rangle. \tag{1.14}$$

(1) En otro caso $\mathrm{proj}(\mathbf{x}; C) = \mathbf{x}$.

Por convenio, si $D = \emptyset$ se toma $\sigma_D$ 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*

$$\langle \mathbf{u}_0,\, \mathbf{x}_0 \rangle = \sigma_C(\mathbf{u}_0). \tag{1.15}$$

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 (1.15),

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

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

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

Recíprocamente, si $\mathrm{proj}(\mathbf{x}_0 + \mu\mathbf{u}; C) = \mathbf{x}_0$ para $\mu > 0$, de la caracterización variacional de la proyección orotogonal (1.12), 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 $\mathrm{proj}(\mathbf{x}_0 + \mu\mathbf{u}; C) = \mathbf{x}_0$, $\mu > 0$.*

**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

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

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

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

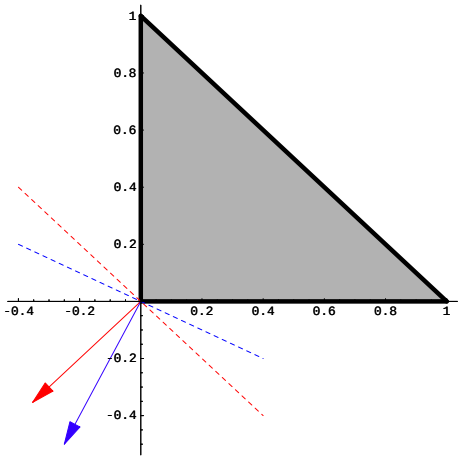



Figura 1.5: Hiperplanos soportando $T$

con $a, b \leq 0$, 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.

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

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

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

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

$$\frac{\partial f}{\partial y_j}(\mathbf{y}) = -2\left(\langle \mathbf{x}_0 + \mu\mathbf{u},\, \mathbf{e}_j\rangle - y_j\right) - 2\left(\langle \mathbf{x}_0 + \mu\mathbf{u},\, \mathbf{e}_N\rangle - \psi(\mathbf{y})\right)\frac{\partial \psi}{\partial y_j}(\mathbf{y}), \quad (1.16)$$

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 $\mathrm{proj}(\mathbf{x}_0 + \mu\mathbf{u}; C) = \mathbf{x}_0$, es decir, si $\mathbf{x}_0$ minimiza $f$ o, equivalentemente, si es un punto crítico, lo que por las ecuaciones (1.16) nos lleva a escribir las condiciones

$$-2\mu u_j - 2\mu u_N \frac{\partial \psi}{\partial y_j}(\mathbf{y}_0) = 0 \;\Leftrightarrow\; u_j = -u_N \frac{\partial \psi}{\partial y_j}(\mathbf{y}_0), \qquad 1 \leq j \leq N-1 \quad (1.17)$$

$\mathbf{y}_0 = (x_{01}, \ldots, x_{0N-1}) \in \mathbb{R}^{N-1}$. 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 $1 \leq j \leq N-1$

$$\frac{\partial \varphi}{\partial x_j}(\mathbf{x}_0) + \frac{\partial \varphi}{\partial x_N}(\mathbf{x}_0) \frac{\partial \psi}{\partial y_j}(\mathbf{y}_0) = 0 \quad (1.18)$$

Combinando (1.17) con (1.18) se llega a la relación

$$u_j = u_N \frac{\partial \varphi}{\partial x_j}(\mathbf{x}_0) / \frac{\partial \varphi}{\partial x_N}(\mathbf{x}_0) \quad (1.19)$$

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.

## 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*

$$\phi\left(\sum_{j=1}^{m} \lambda_j \mathbf{x}_j\right) \leq \sum_{j=1}^{m} \lambda_j \phi(\mathbf{x}_j) \tag{1.20}$$

*La expresión anterior se denomina* desigualdad de Jensen.

Razonando como en el caso de los conjuntos convexos (ver (1.9) en la página 9) 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 $f$

$$\mathrm{gr}(f) = \{(\mathbf{x}, f(\mathbf{x})) : \mathbf{x} \in D\} \tag{1.21}$$

- *Epigráfica* o *epigrafo* de $f$

$$\mathrm{epi}(f) = \left\{(\mathbf{x}, \alpha) \in \mathbb{R}^{N+1} : \mathbf{x} \in D,\ f(\mathbf{x}) \leq \alpha\right\} \tag{1.22}$$

Claramente $\mathrm{gr}(f) \subset \mathrm{epi}(f)$, 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 $\phi$ 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

$$\begin{aligned}
\frac{1}{2}\left|(1-\lambda)\mathbf{x} + \lambda\mathbf{y}\right|^2 &= \frac{1}{2}\left((1-\lambda)^2|\mathbf{x}|^2 + 2(1-\lambda)\lambda\langle\mathbf{x}, \mathbf{y}\rangle + \lambda^2|\mathbf{y}|^2\right) \\
&\leq \frac{1}{2}\left((1-\lambda)^2|\mathbf{x}|^2 + 2(1-\lambda)\lambda|\mathbf{x}|\,|\mathbf{y}| + \lambda^2|\mathbf{y}|^2\right) \\
&= \frac{1}{2}\left((1-\lambda)|\mathbf{x}| + \lambda|\mathbf{y}|\right)^2
\end{aligned} \tag{1.23}$$

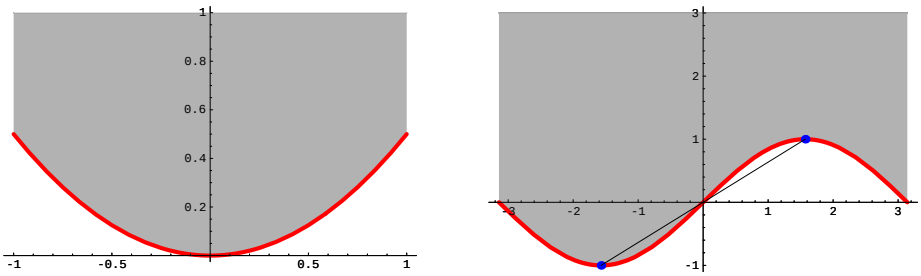
usando además la desigualdad de Cauchy-Schwarz. Por otra parte, dados dos números reales $a, b > 0$, de la fórmula del cuadrado del binomio se tiene que $(a+b)^2 \leq a^2 + b^2$ y si $0 \leq \lambda \leq 1$,

$$((1-\lambda)a + \lambda b)^2 \leq (1-\lambda)^2 a^2 + \lambda^2 b^2 \leq (1-\lambda)a^2 + \lambda b^2 \qquad (1.24)$$

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 1.6 corresponde a la epigráfica de la función anterior para $N = 1$. Se observa que el segmento uniendo dos puntos cualesquiera de la gráfica de $\phi$ (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 1.6, se observa que tomando, por ejemplo, los puntos $(-\pi/2, -1)$, $(\pi/2, 1) \in \mathrm{gr}(\varphi)$, el segmento que los une no está contenido en $\mathrm{epi}(\varphi)$, por lo que esta función no es convexa.




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

Figura 1.6: Epigráficas

**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, $\mathbf{x}^t Q\mathbf{x} \geq 0$, para cada $\mathbf{x} \in \mathbb{R}^N$. Con estas hipótesis se verifica la *desigualdad de Cauchy-Schwarz* que establece que

$$\forall\ \mathbf{x}, \mathbf{y} \in \mathbb{R}^N \qquad \left|\mathbf{x}^t Q\mathbf{y}\right|^2 \leq \left(\mathbf{x}^t Q\mathbf{x}\right)\left(\mathbf{y}^t Q\mathbf{y}\right) \qquad (1.25)$$

Además, si definimos la función

$$\varphi(\mathbf{x}) = \mathbf{x}^t Q\mathbf{x} = \langle \mathbf{x}, Q\mathbf{x}\rangle, \qquad \mathbf{x} \in \mathbb{R}^N \qquad (1.26)$$

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 (1.25) y (1.24), de la identidad anterior se obtiene la desigualdad de Jensen para $\varphi$:

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

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

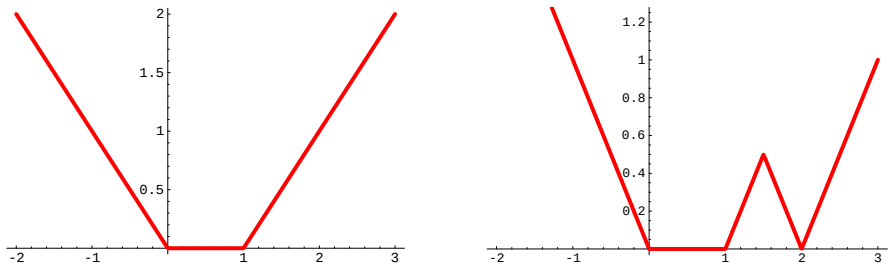
**Ejemplo 1.11 (Funciones distancia)** Dado un conjunto convexo $C \subset \mathbb{R}^N$, la función distancia a $C$, $d_C(\cdot)$, definida por la relación $(1.4)$ es convexa. En efecto, dados $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$, para cada $\varepsilon > 0$ existirán $\mathbf{x}_\varepsilon, \mathbf{y}_\varepsilon \in C$, 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 1.8 corresponde a la gráfica de la función




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

Figura 1.7: Funciones distancia

distancia al conjunto $D = \left\{(x,y) \in \mathbb{R}^2 : (x-1)^2 + (y-1)^2 \leq 1\right\}$. El recíproco de la afirmación anterior también es cierto si $C \subset \mathbb{R}^N$ es un conjunto cerrado.

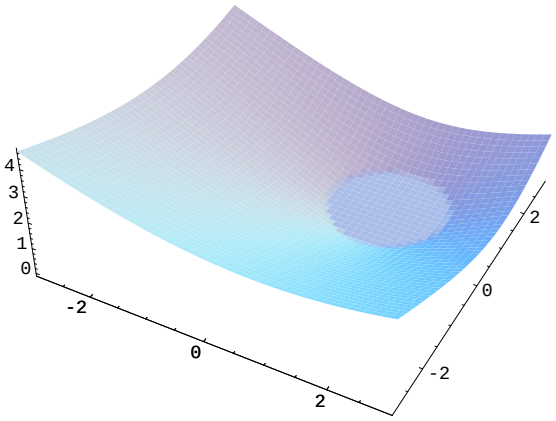



Figura 1.8: Gráfica de $d_D(\cdot)$

En efecto, en ese caso sabemos que $C = \{\mathbf{x} \in \mathbb{R}^N : d_C(\mathbf{x}) = 0\}$, 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* $\text{epi}(\phi) \subset \mathbb{R}^{N+1}$ *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. 22).

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

$$\text{dom}(\phi) = \{\mathbf{x} \in \mathbb{R}^N : \phi(\mathbf{x}) \in \mathbb{R}\}. \tag{1.27}$$

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 de deduce de la identidad

$$\operatorname{epi}\left(\sup_{i\in I}\phi_i\right) = \bigcap_{i\in I}\operatorname{epi}(\phi_i) \tag{1.28}$$

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, $i \in I$. Se tiene que la función*

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

*es convexa.*

**Ejemplo 1.12 (Funciones soporte)** Del corolario anterior es evidente que las funciones soporte $\sigma_D$, definidas en (1.14), son convexas independientemente del conjunto $D$. Además, en el caso de conjuntos no acotados porporcionan ejemplos de funciones que de forma natural toman el valor $+\infty$ en algunos puntos.

### 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 $\phi$ 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 $\phi$ es convexa si y solamente si para cada $\mathbf{x},\mathbf{y} \in D$ se verifica alguna de las siguientes desigualdades:*

$$\langle\nabla\phi(\mathbf{x}),\, \mathbf{y}-\mathbf{x}\rangle \leq \phi(\mathbf{y}) - \phi(\mathbf{x}) \tag{1.29}$$

$$\langle\nabla\phi(\mathbf{y}) - \nabla\phi(\mathbf{x}),\, \mathbf{y}-\mathbf{x}\rangle \geq 0 \tag{1.30}$$

Cuando se verifica (1.30) se dice que el gradiente de $\phi$ es *monótono*. En el caso unidimensional esto significa que la función derivada $\phi'$ 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 $\phi$ 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

$$\begin{aligned} &\text{(i) } \bar{\mathbf{x}} \in K \\ &\text{(ii) } f(\bar{\mathbf{x}}) = \inf_{\mathbf{x} \in K} f(\mathbf{x}) \end{aligned} \tag{1.31}$$

Definiendo la *función indicatriz* del conjunto

$$\psi_K(\mathbf{x}) = \begin{cases} 0, & \text{si } \mathbf{x} \in K \\ +\infty, & \text{si } \mathbf{x} \notin K \end{cases} \tag{1.32}$$

el problema (1.31) 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 $\phi$ 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 $\phi$ 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 $\phi$ es *estrictamente convexa* es decir, si dados $\mathbf{x}, \mathbf{y} \in \mathrm{dom}(\phi)$ dos puntos distintos se tiene que

$$\phi\left(\frac{\mathbf{x}+\mathbf{y}}{2}\right) < \frac{\phi(\mathbf{x}) + \phi(\mathbf{y})}{2} \tag{1.33}$$

En efecto, si $\overline{m}$ es el ínfimo de la función $\phi$, 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 $\phi$ es estrictamente convexa si y solamente si la desigualdad (1.29) o (1.30) se verifica de forma estricta para cada $\mathbf{x} \neq \mathbf{y}$. Si $\phi$ es $C^2$ una condición suficiente, aunque no necesaria, para la convexidad estricta de $\phi$ 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 $\phi$, 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

$$\phi_\lambda(\mathbf{x}) = \inf_{\mathbf{y} \in \mathbb{R}^N} \left( \phi(\mathbf{y}) + \frac{1}{2\lambda} |\mathbf{y} - \mathbf{x}|^2 \right) \tag{1.34}$$

La aplicación $\phi_\lambda \leq \phi$, se denomina *envoltura de Moreau (Moreau envelope)*.

Dado $\mathbf{x} \in \mathrm{dom}(\phi)$, supongamos que $\bar{\mathbf{x}} \in \mathbb{R}^N$ es solución de (1.34), es decir,

$$\phi(\bar{\mathbf{x}}) + \frac{1}{2\lambda} |\bar{\mathbf{x}} - \mathbf{x}|^2 \leq \phi(\mathbf{y}) + \frac{1}{2\lambda} |\mathbf{y} - \mathbf{x}|^2 \tag{1.35}$$

para cada $\mathbf{y} \in \mathbb{R}^N$. 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 $\phi$ 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

$$\phi(\bar{\mathbf{x}}) - \phi(\mathbf{y}) + \frac{1}{\lambda} \langle \mathbf{x} - \bar{\mathbf{x}},\, \mathbf{y} - \bar{\mathbf{x}} \rangle \leq 0, \qquad \forall\, \mathbf{y} \in \mathbb{R}^N \tag{1.36}$$

que necesariamente debe verificar la solución de (1.34). Recíprocamente, si $\bar{\mathbf{x}} \in \mathrm{dom}(\phi)$ verifica (1.36), 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 (1.34).

Una consecuencia inmediata de la caracterización variacional de la solución del problema (1.34) es su unicidad. Así, si suponemos que $\bar{\mathbf{x}}, \bar{\mathbf{z}}$ son soluciones, de (1.36) 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 $\phi$ 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

$$f(\mathbf{x}) \leq \liminf_{\mathbf{z}\to\mathbf{x}} f(\mathbf{z}) = \sup_{\varepsilon>0}\left(\inf_{\mathbf{z}\in B_\varepsilon(\mathbf{x})} f(\mathbf{z})\right) \tag{1.37}$$

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

$$f(\mathbf{x}) \leq \liminf_{m\to\infty} f(\mathbf{x}_m) = \sup_{M>0}\left(\inf_{m\geq M} f(\mathbf{x}_m)\right) \tag{1.38}$$

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,

$$f \text{ es isc } \;\Leftrightarrow\; \mathrm{epi}(f) \subset \mathbb{R}^{N+1} \text{ es cerrado.} \tag{1.39}$$

Combinando la caracterización anterior con la fórmula (1.28) 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 (1.34).

**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*

$$\operatorname*{argmin}_{\mathbf{y} \in \mathbb{R}^N} \left( \phi(\mathbf{y}) + \frac{1}{2\lambda} |\mathbf{y} - \mathbf{x}|^2 \right) \tag{1.40}$$

*tiene una única solución que denotamos por $J_\lambda(\mathbf{x})$. Dicha solución queda determinada por la desigualdad variacional*

$$\phi(J_\lambda(\mathbf{x})) - \phi(\mathbf{y}) + \frac{1}{\lambda} \langle \mathbf{x} - J_\lambda(\mathbf{x}),\, \mathbf{y} - J_\lambda(\mathbf{x}) \rangle \leq 0, \qquad \forall\ \mathbf{y} \in \mathbb{R}^N \tag{1.41}$$

El campo vectorial $J_\lambda : \mathbb{R}^N \longrightarrow \mathbb{R}^N$ se denomina *resolvente* o *proximal*. De la desigualdad variacional (1.41) se deduce que

$$|J_\lambda(\mathbf{x}) - J_\lambda(\mathbf{z})| \leq |\mathbf{x} - \mathbf{z}|, \qquad \forall\ \mathbf{x}, \mathbf{z} \in \mathbb{R}^N \tag{1.42}$$

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

$$\left| \frac{1}{\lambda}\left(\mathbf{x} - J_\lambda(\mathbf{x})\right) - \frac{1}{\lambda}\left(\mathbf{z} - J_\lambda(\mathbf{z})\right) \right| \leq \frac{1}{\lambda} |\mathbf{x} - \mathbf{z}| \tag{1.43}$$

Los anteriores campos vectoriales verifican las desigualdades

$$\left\langle \frac{1}{\lambda}\left(\mathbf{x} - J_\lambda(\mathbf{x})\right) - \frac{1}{\lambda}\left(\mathbf{z} - J_\lambda(\mathbf{z})\right),\, \mathbf{x} - \mathbf{z} \right\rangle \geq \frac{|\mathbf{x} - J_\lambda(\mathbf{x}) - \mathbf{z} + J_\lambda(\mathbf{z})|^2}{\lambda} \tag{1.44}$$

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

de donde se deduce que tanto la regularización de Yosida como la resolvente son campos vectoriales *monótonos*[(2)].

**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

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

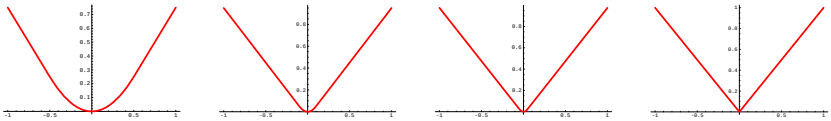
absoluto, $\phi(x) = |x|$, $x \in \mathbb{R}$. Para cada $\lambda > 0$, un análisis detallado permite obtener la expresión explícita de la función (1.34) asociada

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

y del campo resolvente

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

En este ejemplo sencillo se observan diversas propiedades que se probarán más



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

Figura 1.9: Gráficas de $\phi_\lambda$ para $\phi(x) = |x|$

adelante para el caso general. Así, vemos que $\phi_\lambda$ es convexa y de clase $C^1$, con $\phi_\lambda \to \phi$ puntualmente cuando $\lambda \to 0$. Además $J_\lambda(x) \to x$ 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 (1.32). 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 $\mathrm{proj}(\mathbf{x}; C)$ 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 $\phi$ es localmente Lipschitz en el interior de su dominio.*

**Nota 1.3** La prueba del teorema anterior (ver [2], [6]) proporciona información adicional sobre la constante de Lipschitz $\alpha(\mathbf{x})$. En efecto, si $\gamma(\mathbf{x}) > 0$ es una cota de $\phi$ 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

$$|\phi(\mathbf{y}) - \phi(\mathbf{z})| \leq \frac{2\gamma(\mathbf{x})}{\delta/2}|\mathbf{y} - \mathbf{z}| \tag{1.46}$$

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 [16, pág. 83-824] y [17, 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 $\sigma_C$ 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 $\sigma_C$ es continua en $\operatorname{int}(\operatorname{dom}\sigma_C) = \left\{(x, y) \in \mathbb{R}^2 : y > 0\right\}$ (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 $\sigma_C$ 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 $\phi$ 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 $\phi$ 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):

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

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 $\phi$ 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*

$$\langle \mathbf{u}, \mathbf{x}\rangle - \phi(\mathbf{x}) = \sup_{(\mathbf{z},\beta)\in\operatorname{epi}\phi} (\langle \mathbf{u}, \mathbf{z}\rangle - \beta) = \sigma_{\operatorname{epi}\phi}(\mathbf{u}, -1). \tag{1.47}$$

*El conjunto de todos los subgradientes se denomina* subdiferencial *de $\phi$ en $\mathbf{x}$ y se denota por $\partial\phi(\mathbf{x})$. Se dice que $\phi$ 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

$$u - 1 = \sup_{|x|\leq 1,\, \beta \geq x^2} (ux - \beta) = \sup_{|x|\leq 1} \left(ux - x^2\right) \tag{1.48}$$

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 (1.48) 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 (1.48) 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 ([16], 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 $\psi_C$ 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):

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

La denotaremos por $\delta\phi(\mathbf{x}; \mathbf{u})$. Cuando $\phi$ 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 $\partial\phi(\mathbf{x}; \mathbf{u})$ para $|\mathbf{u}| = 1$. Por ejemplo, si $\phi(x) = |x|$ es la función valor absoluto, se tiene que

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

de donde es inmediato comprobar que $\partial\phi(\mathbf{x}; \mathbf{u}) = |\mathbf{u}|$. 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*

$$\partial\phi(\mathbf{x}) = \left\{ \mathbf{u} \in \mathbb{R}^N : \langle \mathbf{u}, \mathbf{v} \rangle \leq \delta\phi(\mathbf{x}; \mathbf{v}),\ \forall\ \mathbf{v} \in \mathbb{R}^N \right\}. \tag{1.49}$$

**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* $\phi$ *es inferiormente semicontinua en* $\mathbf{x}$*.*

*(ii)* $\mathbf{0} \in \partial\phi(\mathbf{x}) \Leftrightarrow \mathbf{x}$ *es un mínimo de* $\phi$ *(*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* $\phi$ *en un entorno de* $\mathbf{x}$*, se verifica:*

$$\sup_{\mathbf{u}\in\partial\phi(\mathbf{x})} |\mathbf{u}| \leq \alpha(\mathbf{x}) \tag{1.50}$$

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

$$\sup_{\mathbf{z}\in B_\varepsilon(\mathbf{x})} \left( \sup_{\mathbf{u}\in\partial\phi(\mathbf{z})} |\mathbf{u}| \right) \leq \alpha(\mathbf{x}) \;\Leftrightarrow\; \partial\phi(\mathbf{z}) \subset \alpha(\mathbf{x})\mathcal{B}, \quad \forall\, \mathbf{z} \in B_\varepsilon(\mathbf{x}) \tag{1.51}$$

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 $\phi$ 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, (1.51) 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

$$\sup_{\mathbf{x}\in K} \left( \sup_{\mathbf{u}\in\partial\phi(\mathbf{x})} |\mathbf{u}| \right) \leq \frac{2\gamma}{\varepsilon/2} \tag{1.52}$$

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

$$\operatorname{Gr}(\partial\phi) = \left\{ (\mathbf{x}, \mathbf{u}) \in \mathbb{R}^{2N} : \mathbf{u} \in \partial\phi(\mathbf{x}) \right\}. \tag{1.53}$$

Si la función $\phi$ 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 o derivadas direccionales, como la función soporte de la subdiferencial.

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

$$\delta\phi(\mathbf{x};\mathbf{v}) = \sigma_{\partial\phi(\mathbf{x})}(\mathbf{v}), \quad \forall\ \mathbf{v} \in \mathbb{R}^N . \tag{1.54}$$

**Teorema 1.9** *Sean* $\phi : \mathbb{R}^N \longrightarrow \mathbb{R} \cup \{+\infty\}$ *estricta y convexa,* $\mathbf{x} \in \operatorname{int}(\operatorname{dom}\phi)$. *Se tiene que* $\phi$ *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 (1.41) y el Corolario 1.4 es evidente que la regularización de Yosida está contenida en el conjunto subdiferencial de $\phi$ en la resolvente, es decir,

$$\frac{1}{\lambda}\left(\mathbf{x} - J_\lambda(\mathbf{x})\right) \in \partial\phi(J_\lambda(\mathbf{x})). \tag{1.55}$$

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

$$\left|\frac{1}{\lambda}\left(\mathbf{x} - J_\lambda(\mathbf{x})\right) - \partial\phi^0(\mathbf{x})\right|^2 \leq |\partial\phi^0(\mathbf{x})|^2 - \left|\frac{1}{\lambda}\left(\mathbf{x} - J_\lambda(\mathbf{x})\right)\right|^2 . \tag{1.56}$$

De aquí, usando (1.50), se llega a

$$|\mathbf{x} - J_\lambda(\mathbf{x})| \leq \lambda|\partial\phi^0(\mathbf{x})| \leq \lambda\alpha(\mathbf{x}) \underset{\lambda\to 0^+}{\longrightarrow} 0 \tag{1.57}$$

es decir, la resolvente converge a la identidad puntualmente en el interior del dominio de $\phi$. Es más, si $K \subset \operatorname{int}(\operatorname{dom}\phi)$ es un compacto, de (1.52), existirá $\alpha > 0$ de forma que

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

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 (1.34), es una función convexa, cuya subdiferencial en cada punto contiene a la regularización de Yosida

$$\frac{1}{\lambda}\left(\mathbf{x} - J_\lambda(\mathbf{x})\right) \in \partial\phi_\lambda(\mathbf{x}). \tag{1.59}$$

Es más, $\phi_\lambda$ es diferenciable con

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

Veamos ahora hacia que converge la envoltura de Moreau. Claramente $\phi_\lambda(\mathbf{x}) \leq \phi(\mathbf{x})$, luego

$$\limsup_{\lambda \to 0^+} \phi_\lambda(\mathbf{x}) \leq \phi(\mathbf{x}). \tag{1.61}$$

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 $\phi$ inferiormente semicontinua podemos escribir

$$\begin{aligned} \phi(\mathbf{x}) &\leq \liminf_{\lambda \to 0^+} \phi(J_\lambda(\mathbf{x})) \leq \liminf_{\lambda \to 0^+} \phi(J_\lambda(\mathbf{x})) + \frac{1}{2\lambda}|J_\lambda(\mathbf{x}) - \mathbf{x}|^2 \\ &= \liminf_{\lambda \to 0^+} \phi_\lambda(\mathbf{x}) \end{aligned} \tag{1.62}$$

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 (1.61)-(1.62) se tiene que existe el límite puntual de la envoltura de Moreau y

$$\lim_{\lambda \to 0^+} \phi_\lambda(\mathbf{x}) = \phi(\mathbf{x}) \tag{1.63}$$

También existe el límite puntual de la regularización de Yosida,

$$\hat{\mathbf{y}}(\mathbf{x}) = \lim_{\lambda \to 0^+} \frac{1}{\lambda} \left(\mathbf{x} - J_\lambda(\mathbf{x})\right) \tag{1.64}$$

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 (1.55), (1.57), (1.64) 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 (1.56). 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

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

## 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

$$-\dot{\mathbf{x}}(t) = \nabla\phi(\mathbf{x}(t)) \tag{1.66}$$

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, [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

$$-\dot{\mathbf{x}}(t) \in \partial\phi(\mathbf{x}(t)). \tag{1.67}$$

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

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

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 con objeto de describir la evolución de conjuntos a lo largo del 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

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

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

$$-\dot{\mathbf{x}}(t) \in \partial\phi(\mathbf{x}(t)) + \beta(t)\partial\varphi(\mathbf{x}(t)) \tag{1.70}$$

donde $\beta : [0, T[ \longrightarrow \mathbb{R}$ es una función. Cabe decir que bajo ciertas hipótesis, por ejemplo,

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

$$\mathbf{0} \in \operatorname{int}(\operatorname{dom}\phi - \operatorname{dom}\varphi). \tag{1.71}$$

- $\beta$ toma valores no negativos.

se tiene la identidad

$$\partial\phi(\mathbf{x}) + \beta(t)\partial\varphi(\mathbf{x}) = \partial\left(\phi + \beta(t)\varphi\right)(\mathbf{x}), \tag{1.72}$$

con lo que (1.70) no es mas que un caso particular de (1.68), a pesar de lo cual sigue teniendo interés (ver p.e. [1]). Sin embargo (1.72) no se da en general, lo que obliga a tratar este problema de forma independiente. Tiene 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].

### 1.4.2 Existencia de solución

**Definición 1.5** *Se denomina* solución en sentido fuerte (strong solution) o de Brézis *de (1.68) 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 (1.67), la noción de solución es análoga, mientras que para el caso perturbado (1.69), 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 (1.70) si para casi todo* $0 < t < T$*,*

---

[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

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

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

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

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

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

$$\mathbf{x}(0) = \mathbf{x}_0. \tag{1.73}$$

Además la solución depende de forma continua de la condición inicial.

► Ecuación (1.67): 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 (1.67) para las condiciones iniciales* $\mathbf{x}_0$, $\mathbf{y}_0$, *respectivamente, se tiene que, para cada* $t > 0$,

$$|\mathbf{x}(t) - \mathbf{y}(t)| \leq |\mathbf{x}_0 - \mathbf{y}_0| \tag{1.74}$$

Obviamente la estimación (1.74) 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 (1.67)+(1.73),* $\mathbf{x}(\cdot)$, *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*

$$\frac{d}{dt}\left(\phi(\mathbf{x}(t))\right) + |\dot{\mathbf{x}}(t)|^2 = 0, \tag{1.75}$$

*de donde se deduce que las soluciones de (1.67) proporcionan trayectorias de descenso de* $\phi$.

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

- *Convergencia asintótica:*

$$\lim_{t\to+\infty} \mathbf{x}(t) \in \operatorname{argmin} \phi \tag{1.76}$$

- *Convergencia ergódica:*

$$\lim_{t\to+\infty} \frac{1}{t}\int_0^t \mathbf{x}(s)ds \in \operatorname{argmin} \phi \tag{1.77}$$

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

$$\begin{cases} -\dot{\mathbf{x}}_\lambda(t) = \nabla\phi_\lambda(\mathbf{x}_\lambda(t)), \\ \mathbf{x}_\lambda(0) = \mathbf{x}_\lambda, \end{cases} \tag{1.78}$$

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 (1.67)+(1.73). Esta es esencialmente la prueba del Teorema 1.11, para los detalles nos remitimos a [3] y [4].

► Ecuación (1.68): potencial variable. Sea $\phi : [0, T[\times\mathbb{R}^N \longrightarrow \mathbb{R}\cup\{+\infty\}$ estricta de forma que

($\phi$1) Para cada $0 < t < T$, $\phi(t, \cdot)$ es convexa e inferiomente semicontinua.

($\phi$2) 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

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

Si $\mathbf{x}(\cdot)$, $\mathbf{y}(\cdot)$ son soluciones de (1.68) para las condiciones iniciales $\mathbf{x}_0$, $\mathbf{y}_0$, respectivamente, se verifica la desigualdad (1.74), lo que proporciona la dependencia continua de las condiciones iniciales y la unicidad para el problema de Cauchy (1.68)+(1.73). En cuanto a la existencia, las condiciones ($\phi$1)-($\phi$2) garantizan que la función $\nabla\phi_\lambda(t, \mathbf{x}) = \frac{1}{\lambda}\left(\mathbf{x} - J_\lambda(t, \mathbf{x})\right)$ es continua respecto de la variable $t$ y Lipschitz respecto de $\mathbf{x}$, por lo que los problemas

$$\begin{cases} -\mathbf{x}_\lambda(t) = \nabla\phi_\lambda(t, \mathbf{x}_\lambda(t)), \\ \mathbf{x}_\lambda(0) = \mathbf{x}_\lambda, \end{cases} \tag{1.79}$$

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 (1.68)+(1.73). 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 (1.68)+(1.73) 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 (1.69): caso perturbado. Supongamos que $\phi$ es un potencial verificando las hipótesis de los apartados anteriores, dependiendo de si depende o no del 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 (1.69) para las condiciones iniciales* $\mathbf{x}_0$*,* $\mathbf{y}_0$*, respectivamente, se verifica la estimación*

$$|\mathbf{x}(t) - \mathbf{y}(t)| \leq |\mathbf{x}_0 - \mathbf{y}_0| \left(e^{\int_0^t \gamma(s)ds}\right)^{1/2}, \quad 0 < t < T. \tag{1.80}$$

De (1.80) es inmediata la dependencia continua respecto de las condiciones iniciales y la unicidad de las soluciones de (1.69) 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 (1.69) para los términos fuente* $\mathbf{F}$ *y* $\mathbf{G}$*, respectivamente, con* $\mathbf{x}_0$*,* $\mathbf{y}_0$ *las condiciones iniciales respectivas. Se tiene entonces*

$$|\mathbf{x}(t) - \mathbf{y}(t)|^2 \leq |\mathbf{x}_0 - \mathbf{y}_0|^2 e^{\int_0^t (1+\gamma(s))ds} + \int_0^t r(s)^2 e^{\int_s^t (1+\gamma(\tau))d\tau} ds \tag{1.81}$$

*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 $\phi$ y el término fuente $\mathbf{F}$ garantizan que el problema

$$\begin{cases} -\mathbf{x}_\lambda(t) = \nabla\phi_\lambda(t,\mathbf{x}_\lambda(t)) + \mathbf{F}(t,\mathbf{x}_\lambda(t)), \\ \mathbf{x}_\lambda(0) = \mathbf{x}_\lambda, \end{cases} \tag{1.82}$$

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 (1.69)+(1.73) definida en $[0,T[$*

► Ecuación (1.70): 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:

($\varphi$1) $\partial\phi(\mathbf{x}) \subseteq \partial\varphi(\mathbf{x})$, para todo $\mathbf{x}$.

($\varphi$2) Existen $0 < k < 1$, $\eta : \mathbb{R} \longrightarrow \mathbb{R}_+$ creciente de forma que, para cada $\mathbf{x} \in \operatorname{dom}\phi$:

$$|\partial\varphi^0(\mathbf{x})| \leq k|\partial\phi^0(\mathbf{x})| + \eta(\phi(\mathbf{x}) + |\mathbf{x}|) \tag{1.83}$$

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

($\varphi$3) Se tiene $c > 0$ tal que

$$\varphi(\mathbf{x}) \leq k\phi(\mathbf{x}) + c \tag{1.84}$$

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

($\varphi$4) 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 $\phi$, para cada $\mathbf{u} \in \partial\varphi(\mathbf{x})$, $\mathbf{v} \in \partial\varphi(\mathbf{y})$ se verifica la desigualdad

$$\langle \mathbf{v} - \mathbf{u}, \mathbf{y} - \mathbf{x} \rangle \leq \rho_r(|\mathbf{x}| + |\mathbf{y}|)\,(\phi(\mathbf{x}) + \phi(\mathbf{y}) + K_r)\,|\mathbf{x} - \mathbf{y}|^2 \tag{1.85}$$

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

Usando ($\varphi$4) puede estimarse la distancia entre dos soluciones de (1.70), 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 (1.70) 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*

$$|\mathbf{x}(t) - \mathbf{y}(t)| \leq M(\tau)|\mathbf{x}_0 - \mathbf{y}_0| e^{\frac{1}{2}\int_0^t |\beta(s)|ds}, \quad 0 \leq t \leq \tau. \tag{1.86}$$

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

$$\begin{cases} -\dot{\mathbf{x}}_\lambda(t) = \nabla\phi_\lambda(\mathbf{x}_\lambda(t)) + \beta(t)\nabla\varphi_\lambda(\mathbf{x}_\lambda(t)), \\ \mathbf{x}_\lambda(0) = \mathbf{x}_\lambda, \end{cases} \tag{1.87}$$

tienen una única solución $\mathbf{x}_\lambda(\cdot)$ definida en $[0,+\infty[$. Además, las hipótesis sobre el segundo potencial, ($\varphi$1)-($\varphi$4), garantizan la convergencia de $\mathbf{x}_\lambda(\cdot)$ hacia una función $\mathbf{x}(\cdot)$ que es solución de (1.70)+(1.73) (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 (1.70)+(1.73) 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 (1.70).

**Teorema 1.18 (Theorem 3.1 en [1])** *Supongamos que* $\operatorname{argmin}\varphi \neq \emptyset$,

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

*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* $\varphi^*$ *es la conjugada de Fenchel-Moreau*[(4)] *del potencial* $\varphi$.

*(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 (1.70):*

*(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 $\varphi$ se define como

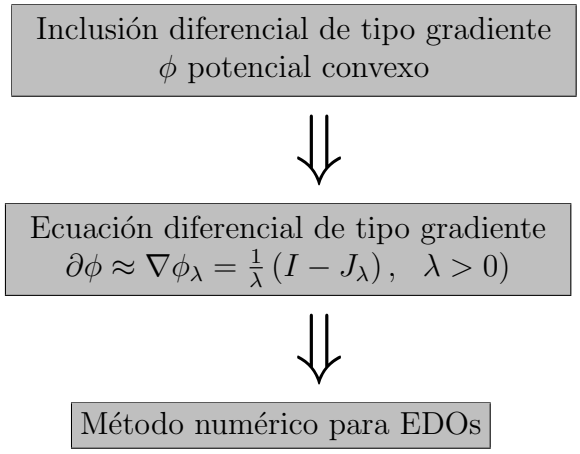
$$\varphi^*(\mathbf{v}) = \sup_{\mathbf{x}\in\mathbb{R}^N}\left(\langle\mathbf{v},\mathbf{x}\rangle - \varphi(\mathbf{x})\right).$$

# Capítulo 2

# Algoritmos

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:




Las soluciones obtenidas en $(\mathcal{AP}_\lambda)$ proporcionan aproximaciones a la solución del problema original $(\mathcal{P})$, de forma que cuando se aumenta la precisión del método numérico y el valor de $\lambda$ se aproxima a cero, mejora el nivel de aproximación.

En este capítulo se presentan diferentes algoritmos que utilizan el método de Moreau-Yosida para aproximar la solución de los problemas expuestos al final del capítulo anterior. 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. Seguidamente pasaremos a comentar y describir cada uno de los códigos generados. Para ello, se comenzará realizando una breve descripción de cada caso seguida de un pseudocódigo y la explicación del código realizado para cada método. Al final de cada apartado de un caso concreto, se expondrán ejemplos para observar el funcionamiento del código implementado en MATLAB.

Cabe decir que los ficheros asociados a los códigos programados en MATLAB se encuentran en el capítulo denominado Anexos.

## 2.1 Métodos para la resolución de ecuaciones diferenciales

Aunque se trata de contenidos estándar, comenzaremos comentado brevemente los algoritmos de Euler y RK4 para resolver de forma aproximada ecuaciones y sistemas de ecuaciones diferenciales. Para una información más detallada puede consultarse cualquier texto de análisis numérico, [11], [13], por ejemplo, o [8] para su implementación en MATLAB.

### 2.1.1 Algoritmo de Euler

El método de Euler es el mas elemental y consiste en sustituir la derivada de la función incógnita por un cociente incremental. Usaremos el denominado Euler explícito,

$$\dot{\mathbf{y}}(t_j) \approx \frac{\mathbf{y}(t_{j+1}) - \mathbf{y}(t_j)}{h_m}$$

que porporciona la recurrencia $\mathbf{y}(t_{j+1}) \approx \mathbf{y}(t_j) + h_m \boldsymbol{f}(t_j, \mathbf{y}(t_j))$ para aproximar las soluciones de

$$\dot{\mathbf{y}}(t) = \boldsymbol{f}(t, \mathbf{y}(t)) \tag{2.1}$$

en los distintos nodos. Los ficheros donde se tiene implementado dicho algoritmo se denominan *euler.m* para el caso de una variable y *eulersist.m* para dos variables.

### 2.1.2 Algoritmo de RK4

Los métodos conocidos como Runge-Kutta fueron introducidos entre 1895 y 1901 por los matemáticos alemanes C.D. Runge, K. Heun y M.W. Kutta como una mejora del método de Euler. Se trata de métodos de un paso (cada nodo depende únicamente del anterior) que son, especialmente el de orden cuatro, ampliamente utilizados por su precisión. La idea del RK4 para aproximar la solución de un problema genérico de la forma (2.1) consiste en usar la representación integral de la solución

$$\mathbf{y}(t) = \mathbf{y}(0) + \int_0^t \boldsymbol{f}(s, \mathbf{y}(s))\, ds$$

y aproximar la integral del término de la derecha mediante un método de cuadratura, de forma que proporcione una aproximación del mayor orden posible. En el caso del método de Runge-Kutta de cuarto orden se tiene la fórmula de recursión

$$\mathbf{y}_{j+1} = \mathbf{y}_j + \frac{h_m}{6}\left(\boldsymbol{f}_{1,j} + 2\boldsymbol{f}_{2,j} + 2\boldsymbol{f}_{3,j} + \boldsymbol{f}_{4,j}\right)$$

con

$$\begin{aligned}
\boldsymbol{f}_{1,j} &= \boldsymbol{f}(t_j, \mathbf{y}_j) \\
\boldsymbol{f}_{i,j} &= \boldsymbol{f}\left(t_j + \frac{h_m}{2}, \mathbf{y}_j + \frac{h_m}{2}\boldsymbol{f}_{i-1,j}\right), \quad i = 2,3 \\
\boldsymbol{f}_{4,j} &= \boldsymbol{f}(t_j + h_m, \mathbf{y}_j + h_m \boldsymbol{f}_{3,j})
\end{aligned}$$

Los códigos asociados se denominan *rk4.m* para una variable y *rk4sistemas.m* para dos.

## 2.2 Algoritmo básico

Consideramos en primer lugar el problema básico (1.67)+(1.73) de determinar el flujo asociado al gradiente de un potencial convexo no necesariamente diferenciable a partir de una condición inicial. En este caso $(\mathcal{P}_\lambda)$ es de la forma

$$\left.\begin{aligned} -\dot{\mathbf{x}}_\lambda(t) &= \tfrac{1}{\lambda}\left(\mathbf{x}_\lambda(t) - J_\lambda(\mathbf{x}_\lambda(t))\right) \\ \mathbf{x}_\lambda(0) &= \mathbf{x}^0 \end{aligned}\right\} \qquad (\mathcal{P}_\lambda)$$

con $J_\lambda(\cdot)$ el campo resolvente. Aplicamos a este problema los métodos de aproximación de Euler y RK4 para obtener $(\mathcal{AP}_\lambda)$.

### 2.2.1 Algoritmo 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:

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

**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**.

---

La principal peculiaridad del algoritmo de Euler a la hora de su implementación es que en el segundo paso se evalúa la resolvente de Moreau de la función $\phi$ en cada nodo $\mathbf{x}_\lambda^{m,j}$, lo que exige calcular el mínimo de la función $\mathbf{y} \rightsquigarrow \phi(\mathbf{y}) + \frac{1}{2\lambda}|\mathbf{y} - \mathbf{x}_\lambda^{m,j}|^2$. Al programar el algoritmo en MATLAB se usa la subrutina `fminserch` para calcular este mínimo. Dicha subrutina, que exige conocer explícitamente la expresión de $\phi$ para poder evaluarla, se basa en el algoritmo Nelder-Mead de búsqueda de extremos mediante un proceso de generación de simplex y evaluación de la función objetivo en sus vértices, que no requiere diferenciabilidad. Además se trata de un procedimiento efectivo que, aunque en general proporciona mínimos locales, en el caso de funciones convexas localiza el mínimo global, pueden consultarse más detalles en [8] y [13].

• **Descripción del código (una variable).** La función denominada `eulerB.m` resuelve problemas de una variable con el método de Euler para el caso básico. Tiene como argumentos los extremos derecho e izquierdo del intervalo, la condición inicial y el número de nodos. Cabe destacar que la función potencial se definirá dentro de un fichero `.m` denominado `phi2.m` sumada con la perturbación correspondiente al cálculo de la resolvente. Por tanto el fichero `phi2.m` posee la función a la que debemos buscar su mínimo y no será necesario pasarla como argumento de entrada. Como argumento de salida se tiene la matriz denominada `E1`.

A la hora del acceso general a ciertos parámetros del programa desde cualquier fichero `.m`, se hace necesario el uso de la herramienta de MATLAB `global`. Con la cual podremos definir variables que serán globales en todo nuestro algoritmo. En este caso definimos como globales las variables `lambda` y `cx`. Donde `lambda` es el parámetro que indica el nivel de aproximación del problema regularizado y `cx` representa cada una de los valores que va tomando la coordenada de la variable `x`.

Con relación a la estructura del algoritmo, tenemos que inicialmente se define el parámetro `h`, el cual viene dado por el cociente de la diferencia de los extremos derecho e izquierdo del intervalo y el número de pasos. A continuación se crean dos vectores de tamaño `M` para la definición del vector de tiempos y el relativo

a la variable `x` denominado `X`. Con ello se define el vector de tiempo como un vector que posee valores desde el extremo inferior del intervalo hasta el extremo superior con paso `h`. A continuación a la primera posición del vector de la variable `x` se le asigna la condición inicial $x_0$ del problema.

Dentro de un bucle *for* se realiza un barrido con M iteraciones en total, donde dentro de éste se irá actualizando el valor de la posición siguiente del vector `X`. Al comienzo del bucle *for*, se asigna a `cx` el valor que poseerá dependiendo de la iteración. Si se trata de la primera iteración a `cx` se le asigna el valor de la condición inicial. Si por el contrario se trata de cualquier otra iteración, a `cx` se le asigna el valor que posea la posición del vector `X` en dicha iteración.

A continuación, tal y como se explicó anteriormente en la descripción de la resolvente, se debe encontrar el mínimo de la función definida en `phi2.m`mediante la subrutina `fminsearch.m`, cuyos argumentos de entrada deben ser la función a la cual queremos buscar su mínimo y el punto donde se comenzará a buscarlo, en nuestro caso será el valor que posea `cx`. Dicha función nos devolverá el punto donde se encuentra el mínimo y el valor de la función en dicho el punto. En nuestro caso solo nos interesa el valor del punto donde se alcanza el mínimo, por tanto solo haremos uso de la variable denominada `minimo`. Una vez calculado dicho mínimo para la iteración `j`, lo guardaremos en nuestro vector creado inicialmente `Min`.

Finalmente se realiza la actualización del vector `X` asignando al valor de la posición siguiente la diferencia entre el valor de la posición anterior y una expresión dependiente del cociente entre los parámetros `h` y `lambda` por la diferencia del valor de la posición anterior y el valor del mínimo en esa misma iteración.

Una vez finalizado dicho barrido se obtendrá un vector `X` solución. La solución del problema se presenta como el vector de tiempos y el de la variable `x` traspuestos.

• **Descripción del código (dos variables).** La función `EulerB2.m` resuelve problemas de dos variables con el método de Euler para el caso básico. Tiene como argumentos los extremos derecho e izquierdo del intervalo, las condiciones iniciales para las dos variables y el número de pasos. Como argumentos de salida se tienen las matrices `E1` y `E2`.

En este programa se definirán tres parámetros como globales: `lambda`, `cx` y `cy`, donde `cy` representa lo mismo que `cx` pero para el caso de la segunda variable.

Las inicializaciones son las mismas que para el caso de una variable, con la excepción de la adición de la creación de un vector para la variable `Y`, al igual que se debe asignar a la primera posición del vector de la variable `Y` la segunda coordenada de la condición inicial $y_0$.

Una vez dentro del bucle *for*, si se trata de la primera iteración, se asigna a `cx` y a `cy` los valores de la primera y segunda condición inicial respectivamente. Si no es así, se le asigna el valor de los vectores `X` e `Y` que posea en dicha iteración. En la búsqueda del mínimo, cabe destacar que solo varía el punto donde se comenzará a buscar, debido a que se trata de una función de dos variables. A

la hora de pasarle el punto de comienzo a la función `fminsearch.m`, debemos hacerlo como un vector por ello se hace uso de unos corchetes. Con esto se obtendrá un valor de un mínimo con dos componentes, una relativa a la variable `x` y la otra a la variable `y`. En este caso no se irán guardando en ningún vector, sino que se irán utilizando para las actualizaciones de las variables y después se machacará su valor por el nuevo.

Por último se actualizan los valores de los vectores correspondientes a las dos variables, de forma que el valor de la posición siguiente será la diferencia entre el valor de la posición anterior y una expresión dependiente del cociente entre los parámetros `h` y `lambda` por la diferencia del valor de la posición anterior y el valor del mínimo en esa misma iteración.

Las soluciones se presentan en dos matrices donde en cada una de ellas se tienen dos vectores: el vector de tiempos y el de la variable correspondiente traspuestos.

### 2.2.2 Algoritmo de RK4

Si usamos el método RK4 para aproximar numéricamente la solución de $(\mathcal{P}_\lambda)$ tenemos 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:

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

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

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

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

  donde

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

  **3.** Definición del nuevo nodo:

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

**4.** Actualización: $j = j + 1$

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

---

• **Descripción del código (una variable).** La función `rk4_B.m` resuelve problemas de una variable con el método de Runge-Kutta 4 para el caso básico. Se hará uso de varios ficheros `.m` para la agilización y ahorro de código. Por ello comenzaremos describiendo y comentando la funcionalidad de cada uno de los archivos utilizados y finalmente el principal `rk4_B.m`.

♢ FUNCIÓN `yo.m`:

Hace referencia a la *resolvente de Yosida* para una variable. Dicha función tiene como argumento de entrada un escalar y como argumento de salida otro escalar. En ella se definen dos parámetros como globales: `lambda` y `minimox`. Se utiliza para el cálculo de los parámetros $k_i$, por lo tanto devuelve lo siguiente:

```
p = -(x(1)-minimox)/lambda
```

donde `x(1)` es el parámetro de entrada proporcionado.

♢ FUNCIÓN `calculamin.m`:

Calcula el mínimo de la función potencial mas la perturbación asociada al cálculo de la resolvente (`phi2.m`), pasándole como argumento el punto donde se comenzará a buscarlo. Para ello llama a la función `fminsearch.m` de MATLAB. Este fichero solo se utiliza para evitar el exceso de líneas repetidas.

♢ FUNCIÓN `rk4_B.m`:

La función denominada `rk4_B.m` resuelve problemas de una variable con el método RK4 para el caso básico. Los argumentos de entrada son los extremos derecho e izquierdo del intervalo,la condición inicial $x_0$ y el número de pasos `M`. Como argumento de salida se tiene la matriz denominada `RK1`. Cabe destacar que en el Runge-Kutta se definen dos nuevas variables globales, `minimox`, donde se irán almacenando de forma temporal los valores del mínimo que vayamos calculando y `puntox`, necesario a la hora de calcular los parámetros para la actualización del vector de la variable `x`, ya que representa el punto en el cual debemos empezar a buscar el mínimo para cada parámetro $k_i$.

En este programa se inicializan los mismos parámetros que en el caso del Euler básico de una variable. Y dentro del bucle *for* sólo varía la parte perteneciente a la actualización del vector de la variable `x`. Con relación a dicha parte, se irá asignando a la variable `puntox` el punto en el cual se desea que comience la búsqueda del mínimo con ayuda de la función ya descrita `calculamin.m`. Inicialmente se le asigna a `puntox` el valor de `cx`,

se calcula el mínimo y con ello se halla el valor de $k_1$ evaluando la función Yosida en el `puntox` y con el valor del mínimo calculado. De esta forma se van calculando cada uno de los valores de los $k_i$ modificando en cada caso el valor de `puntox`. Para $k_2$ `puntox` tendrá un valor igual a $cx+(h*k_1)/2$, para $k_3$ `puntox` valdrá $cx + (h * k_2)/2$ y finalmente para $k_4$, `puntox` será $cx + h * k_3$.

Para cada iteración se irá obteniendo el valor correspondiente a la posición `j` para la construcción del vector `X`. Dicha actualización vendrá dada por la suma del valor de la posición actual mas el sexto de la suma de $k_1$, $2 * k_2$, $2 * k_3$ y $k_4$.

Una vez finalizado dicho barrido se obtendrá un vector `X` solución. La solución del problema se presenta como el vector de tiempos y el de la variable `Y` traspuestos.

• **Descripción del código (dos variables).** La función denominada `rk4_B2.m` resuelve problemas de dos variables con del tipo (1.67) usando el método de Runge-Kutta 4 para aproximar la solución del problema regularizado. Aquí también realizaremos un desglose explicando cada uno de los ficheros .m utilizados, finalizando con el principal.

♢ FUNCIÓN `yo2.m`:

Define la resolvente de Yosida de dos variables. Dicha función tiene como argumento de entrada un vector de tamaño dos y como argumento de salida otro vector de dimensión dos. En ella se define dos parámetros como globales: `lambda` y `minimo`, donde este último tiene dos componentes. Dicha función se utiliza para el cálculo de los parámetros $k_i$ usados en el método Runge-Kutta. Donde cada componente del vector de salida tienen la siguiente expresión:

$$yo\_1 = -\frac{1}{\lambda} * (x(1) - minimo(1));$$
$$yo\_2 = -\frac{1}{\lambda} * (x(2) - minimo(2));$$
$$p = [yo_1, yo_2];$$

donde `x(1)` y `x(2)` son las componentes relativas al vector utilizado como parámetro de entrada.

♢ FUNCIÓN `calculamin2.m`:

Es idéntica a `calculamin.m` en cuanto a funcionalidad y código, con la única diferencia de que el argumento que indica a `fminsearch.m` el punto donde comenzar a buscar el mínimo es un vector de dos componentes.

♢ FUNCIÓN `rk4_B2.m`:

Esta función resuelve problemas de dos variables con el método de Runge-Kutta 4 para el caso básico. Los argumentos de entrada son los mismos que para el caso de una variable, con la salvedad de que se añade una condición inicial $y_0$ para la variable `y`. Como argumento de salida tiene dos matrices, una para cada variable utilizada, denominadas `RK1` y `RK2`.

Como parámetros globales se definen: `lambda`, `minimo`, `puntox` y `puntoy`. Donde `minimo` tiene la misma funcionalidad que `minimox` en `rk4_B.m`. Y donde los parámetros `puntox` y `puntoy` nos indican las coordenadas `x` e `y` del punto en el cual debemos empezar a buscar el mínimo para cada parámetro $k_i$.

Se inicializan los mismos parámetros que en el caso de una variable, pero añadiendo un vector para la segunda variable denominado `Y`. Además se asigna a la primera posición del vector `Y` la segunda condición inicial pasada como argumento de entrada.

Dentro del bucle *for* de `M` iteraciones, inicialmente al igual que en el Euler de dos variables,se asigna a `cx` y a `cy` los valores de la primera y segunda condición inicial respectivamente si se trata de la primera iteración. Si no es así, se le asigna el valor de los vectores `X` e `Y` que posea en dicha iteración. Seguidamente se van calculando los valores de los 4 coeficientes denominados $k_i$, donde $i = 1...4$. Para ello, se irá asignando a las variables `puntox` y `puntoy` las coordenadas deseadas del punto en el cual se desea que comience la búsqueda del mínimo con ayuda de la función *calculamin2.m*. El valor de cada $k_i$, se irá obteniendo gracias a la evaluación de la función Yosida para dos variables. De esta forma se van calculando cada uno de los valores de los $k_i$ modificando en cada caso el valor de `puntox`, `puntoy` de la siguiente manera:

- Para $k_1 \Longrightarrow$ `puntox = cx +` $(h * k_1$ `)/2`
- Para $k_2 \Longrightarrow$ `puntox= cx +` $(h * k_1)$`/2`
- Para $k_3 \Longrightarrow$ `puntox= cx +` $(h * k_2)$`/2`
- Para $k_4 \Longrightarrow$ `puntox= cx +` $h * k_3$

Para cada iteración se irá obteniendo un valor para cada $k_i$ y con ello se podrá obtener el valor correspondiente a la posición `j+1` para la construcción de los vectores relativos a las dos variables `x` e `y`. La actualización de las dos variables vendrá dada por las siguientes expresiones:

$$X(j+1) = X(j) + \frac{h}{6}(k_1(1) + 2k_2(1) + 2k_3(1) + k_4(1))$$
$$Y(j+1) = Y(j) + \frac{h}{6}(k_1(2) + 2k_2(2) + 2k_3(2) + k_4(2))$$

Una vez finalizado el barrido del bucle for se obtendrán dos vectores como solución. El primero de ellos hace referencia a la variable `x` y el segundo a la variable `y`.

### 2.2.3 Ejemplos de aplicación: caso básico

Mostramos a continuación los resultados obtenidos aplicando los códigos anteriormente descritos a diversos problemas concretos.

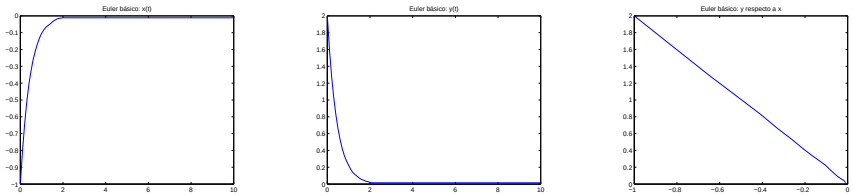
Desde el punto de vista computacional, a la hora de ejecutar los códigos se hace necesario la creación de un fichero `.m` adicional para inicializar los parámetros necesarios, llamar a la función potencial del problema, según el método y número de variables y, finalmente, obtener las salidas (representaciones gráficas, ficheros de datos, etc.) que estimemos oportunas.

Se seguirán los siguientes pasos:

1. Inicialización de los parámetros de entrada de la función a utilizar.
2. Si se utilizaran varios valores de `M`, se realiza un barrido en el cual se inicializa el parámetro `h` y se llama a la función o funciones deseadas.
3. Definición de las representaciones gráficas de la solución proporcionada según el número de variables. Para una variable se realizará solo una: la de la variable respecto al tiempo. Para dos variables se realizarán tres: las de cada variable con respecto al tiempo y la de la curva $(x(t), y(t))$.

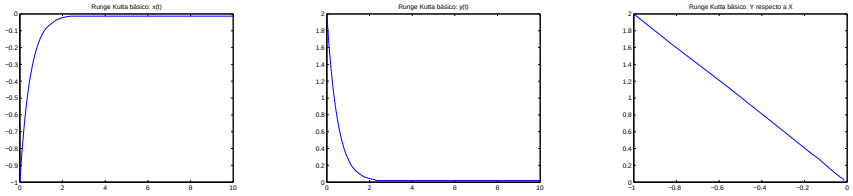
**Ejemplo 2.1** Se considera el sistema asociado al potencial $\phi(x, y) = x^2 + y^2$ y se calcula su solución en el intervalo $[0, 10]$ a partir del valor inicial $(-1, 2)$ mediante los algoritmos de Euler y Runge-Kutta. En este caso la solución exacta puede calcularse de forma analítica, lo que nos permite valorar la exactitud de los algoritmos.

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| `a` | `0` | `lambda` | `0.001` |
| `b` | `10` | `x0` | `-1` |
| `M` | `100` | `y0` | `2` |



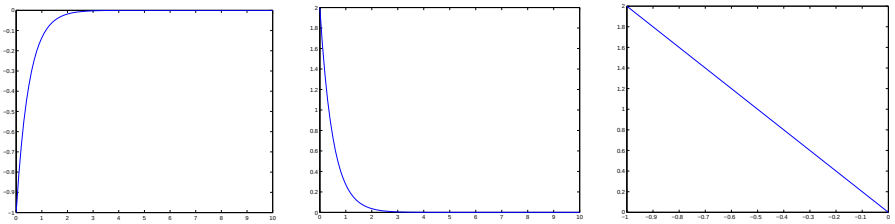

$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.1: Euler básico, $\phi(x, y) = x^2 + y^2$




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

Figura 2.2: Runge-Kutta básico, $\phi(x, y) = x^2 + y^2$



$x(t) = -e^{-2t}$ $y(t) = 2e^{-2t}$ $(x(t), y(t))$

Figura 2.3: Solución exacta, $\phi(x, y) = x^2 + y^2$

Como se puede observar en las imágenes generadas, los resultados obtenidos con ambos métodos son prácticamente idénticos, y a su vez coinciden con los valores exactos de la solución.

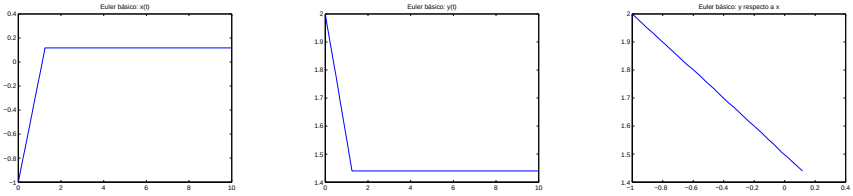
**Ejemplo 2.2** Se considera ahora como potencial la función distancia al círculo $C = \{(x, y) \in \mathbb{R}^2 : (x-1)^2 + (y-1)^2 \leq 1\}$, es decir,

$$\phi(x, y) = d_C(x, y) = \begin{cases} 0, & \text{si } (x, y) \in C \\ -1 + \sqrt{(x-1)^2 + (y-1)^2}, & \text{en otro caso} \end{cases}$$

Volvemos a comparar las soluciones obtenidas mediante los métodos de Euler y RK4 para los parámetros indicados en la siguiente tabla.

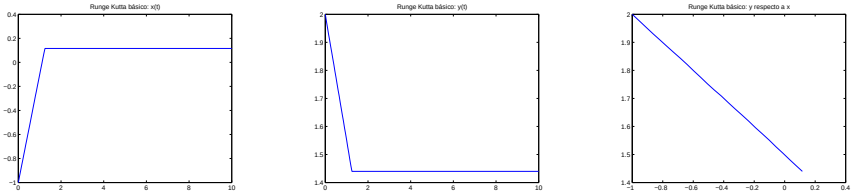
| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 10 | x0 | -1 |
| M | 200 | y0 | 2 |

En las gráficas anteriores observamos que, de acuerdo con el Teorema 1.11, la solución de la inclusión diferencial sigue una dirección de descenso, en este caso se acerca al conjunto $C$ de la forma más rápida (siguiendo la dirección normal).




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

Figura 2.4: Algoritmo Euler básico




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

Figura 2.5: Algoritmo Runge-Kutta básico

## 2.3 Potenciales variables

Puede considerarse el caso en que el potencial depende del tiempo, es decir, se tiene la inclusión diferencial

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

con $\phi : [0, T[ \times \mathbb{R}^N \longrightarrow \mathbb{R}$ convexa en la segunda variable (ver página 36 y siguientes en el capítulo anterior para más detalles).

### 2.3.1 Algoritmo de Euler

La adaptación del algoritmo de Euler es sencilla y solamente hay que modificar el segundo paso que quedará de la forma:

---

**Modificación Euler (potencial variable)**

---

**2.** Cálculo de la resolvente:

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

---

• **Descripción del código (una variable).** La función `eulerC.m` resuelve problemas de una variable con el método de Euler para el caso de potenciales

variables. Esta función posee los mismos argumentos que para el caso básico: los extremos derecho e izquierdo del intervalo,la condición inicial y el número de pasos. De nuevo se necesita un fichero m denominado `phi2.m` análogo al del caso anterior.

Tal y como se ha comentado anteriormente, el caso para potenciales variables será similar al básico con dos excepciones. La primera de ellas es que ahora se tiene un parámetro global adicional, denominado `t`. Será necesario a la hora de evaluar fuera de la función principal debido a que este problema va variando dependiendo del tiempo. La segunda se da en el cálculo del mínimo, ya que antes de calcularlo hay que asignar al parámetro `t` el valor que posea el vector `T` en la iteración en la que se encuentre el bucle. Por tanto de este modo, siendo `t` global, se podrá evaluar en nuestra función `phi2`, la cual dependerá de la variable `x` y del tiempo.

En cuanto a lo demás, el programa posee la misma funcionalidad. Finalmente, se presentará la solución en una matriz como en el caso básico.

• **Descripción del código (dos variables).** El fichero `eulerC2.m` resuelve problemas de dos variables con el método de Euler para el caso de potenciales variables. Se trata del mismo caso de una variable pero extendida para dos, es decir, que posee la misma funcionalidad que en el caso básico, añadiéndole un nuevo parámetro global llamado `t` y evaluando la función con dicho parámetro antes de calcular el mínimo.

### 2.3.2 Algoritmo de RK4

En cuanto al algoritmo Runge-Kutta de orden cuatro, se adapta modificando también el segundo paso en el que se calculan los coeficientes de la actualización, teniendo en cuenta la dependencia temporal del potencial.

---

**Modificación RK4 (potencial variable)**

---

**2.** Cálculo de los coeficientes:

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

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

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

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

donde

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

• **Descripción del código (una variable).** La función denominada `rk4_C.m` resuelve problemas de una variable usando el método de Runge-Kutta 4 para el caso de potenciales variables. En este caso también se hará uso de varios ficheros .m, donde `yo.m` y `calculamin.m` son los mismos que para el caso básico.

En este caso ocurre lo mismo que en en de Euler, es decir, la estructura del programa será la misma que para el caso básico, exceptuando la adición de un nuevo parámetro global denominado `t`, además de la evaluación de la función bajo estudio antes de buscar su mínimo. Debido a que se realizan varias búsquedas de mínimo para ir calculando cada uno de los parámetro $k_i$, se deberá ir actualizando el valor del parámetro `t` además del valor de `puntox`. De este modo, dependiendo del parámetro que estemos calculando, `t` tendrá un valor u otro, que a su vez poseerá dependencia de la iteración en la cual se encuentre el bucle *for*.

Finalmente, la actualización de la variable `x` se realiza como en el método básico y se presenta el resultado en forma de matriz.

• **Descripción del código (dos variables).**

La función `rk4_C2.m` resuelve problemas de dos variables con el método de Runge-Kutta 4 para el caso de potenciales variables. En este caso se usarán varios ficheros .m, donde `yo2.m` y `calculamin2.m` son los mismos que para el caso básico.

Posee la misma estructura que el programa básico, pero con las características para potenciales variables. Es por ello que también se le añade el parámetro global `t`, el cual se irá actualizando para su evaluación en la función para posteriormente realizar el cálculo del mínimo. Con ello se obtendrán los valores de los parámetros $k_i$, los cuales en este caso serán vectores de tamaño dos. Debido a esto, a la hora de actualizar nuestros vectores `X` e `Y`, se deberá seleccionar la primera coordenada de los parámetros $k_i$ para la variable `x` y la segunda para la variable `y`. Dicha actualización es la misma que hemos estado utilizando hasta ahora. Finalmente la solución se presenta en dos matrices, donde cada una de ellas hace referencia a cada una de las dos variables.

### 2.3.3 Ejemplos de aplicación: Potenciales variables

Se muestran a continuación una serie de ejemplos en los que se has usado los algoritmos anteriormente descritos para aproximar la solución de problemas asociados a potenciales convexos variables, (1.68).

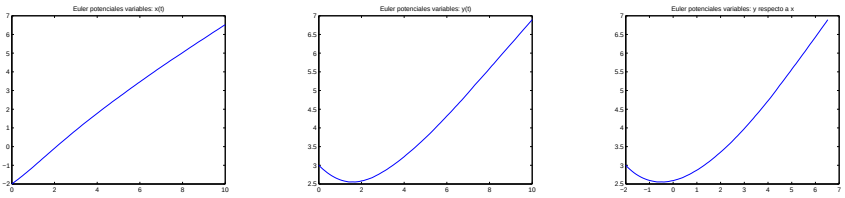
**Ejemplo 2.3** Dada una familia $C(t),\ t > 0$, de conjuntos convexos que varían a lo largo del tiempo, la función distancia, $d_{C(t)}(\mathbf{x})$, constituye un caso particular de potencial convexo dependiendo del tiempo.

Consideramos el caso particular de la familia de círculos con centro y radio variable

$$C(t) = \left\{(x, y) \in \mathbb{R}^2 : (x - (1 + t))^2 + (y - (1 + t))^2 \leq (2 + \cos t)^2\right\}$$

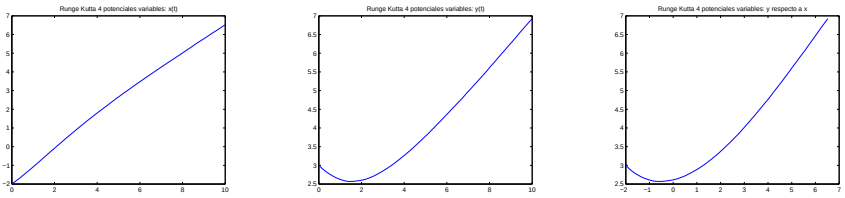
y usamos los algoritmos de Euler y RK4 para aproximar la solución de $-\dot{\mathbf{x}}(t) \in \partial d_{C(t)}(\mathbf{x}(t))$ a partir de los puntos iniciales $(2-,3)$ y $(0,0)$:

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 10 | x0 | -2 |
| M | 100 | y0 | 3 |




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

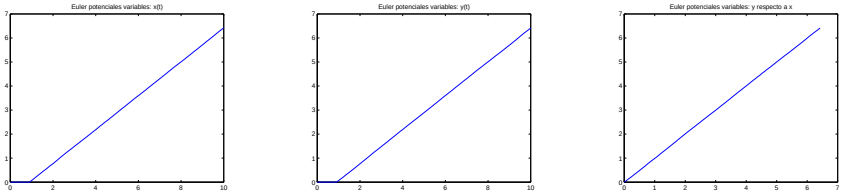
Figura 2.6: Algoritmo de Euler, potencial $d_{C(t)}(x, y)$




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

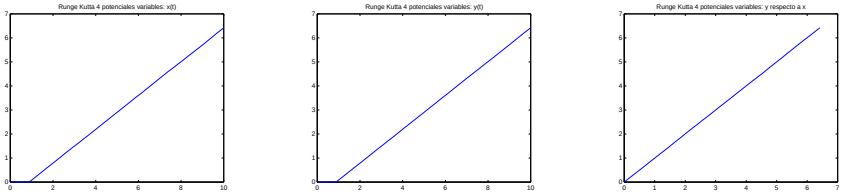
Figura 2.7: Algoritmo RK4, potencial $d_{C(t)}(x, y)$

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 10 | x0 | 0 |
| M | 100 | y0 | 0 |




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

Figura 2.8: Algoritmo Euler, potencial $d_{C(t)}(\mathbf{x})$




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

Figura 2.9: Algoritmo RK4, potencial $d_{C(t)}(\mathbf{x})$

**Ejemplo 2.4** Sea la función,

$$\phi(t, x, y) = \max\left(\sin(10t)(x^2 + y^2), \cos(t)(2x - y + 5)\right) \tag{2.3}$$

que representamos gráficamente para diversos valores del tiempo en la Figura 2.10.

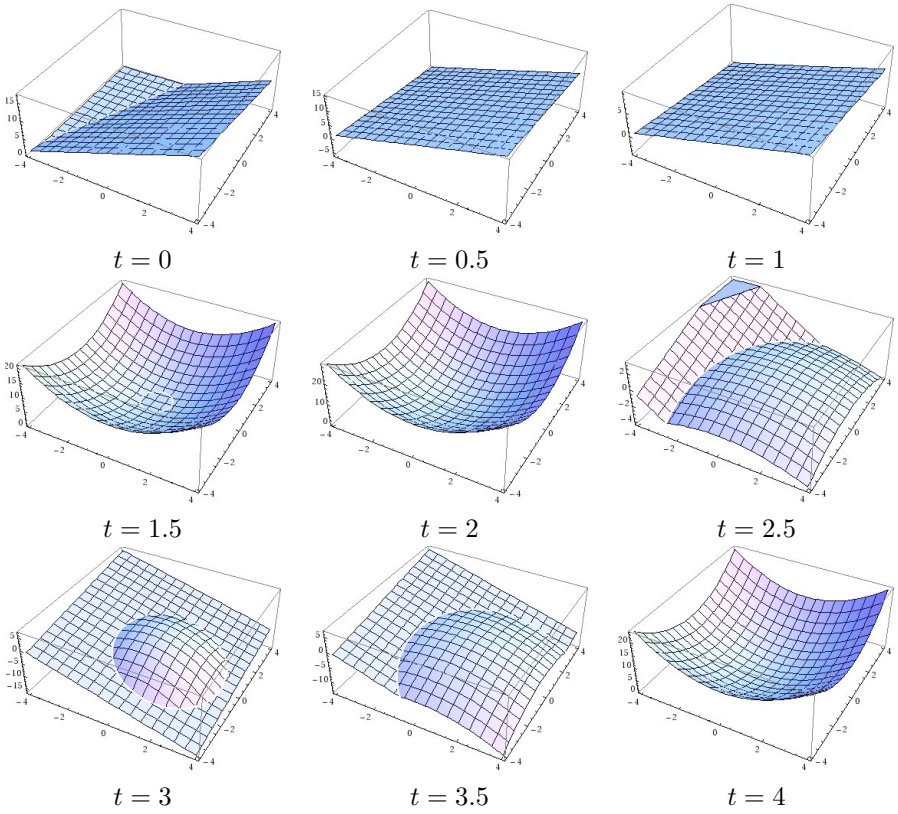
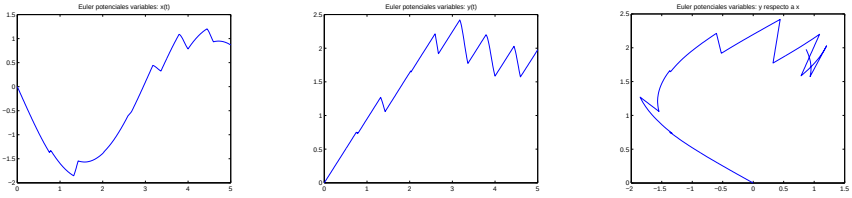



Figura 2.10: Potencial variable $\phi(t, x, y)$

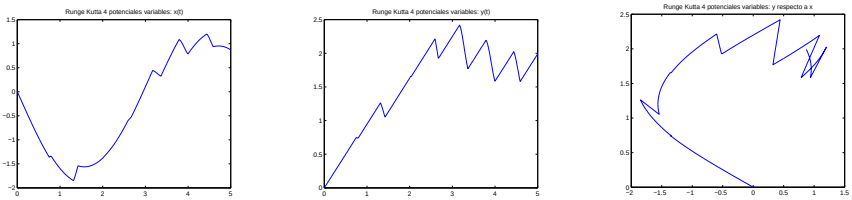
En las Figuras 2.11 y 2.12 se muestran las gráficas de las soluciones proporcionadas por los algoritmos de Euler y Runge-Kutta, respectivamente, para el problema (1.68) en el caso particular del potencial (2.3).

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 5 | x0 | 0 |
| M | 1000 | y0 | 0 |




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

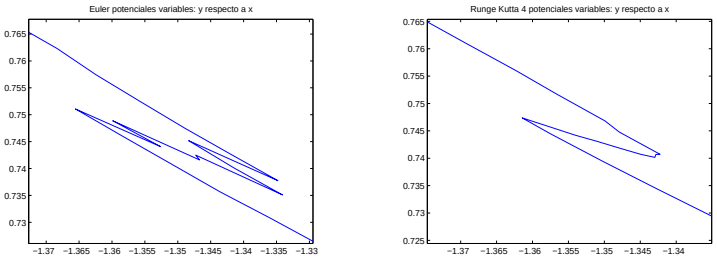
Figura 2.11: Algoritmo de Euler, $\lambda = 0.001$



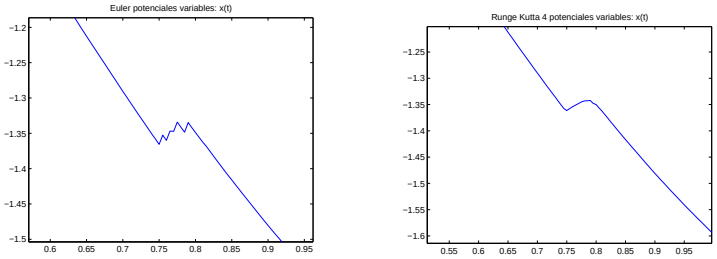

$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.12: Algoritmo de Runge-Kutta, $\lambda = 0.001$

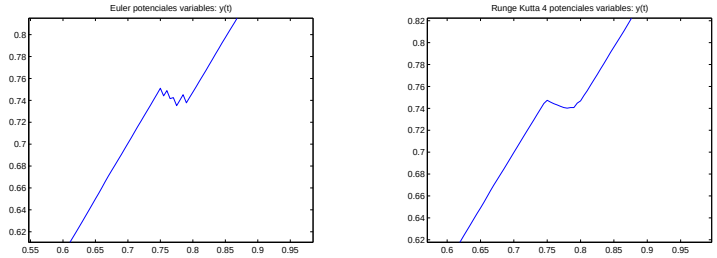
En este ejemplo se observa que la aproximación porporcionada por el método de Euler presenta más oscilaciones que la obtenida aproximando el problema regularizado por el método de Runge-Kutta de cuarto orden. Este hecho se aprecia con más detalle si realizamos una amplicación, así en la Figura 2.13 se muestran detalles de las aproximaciones de Euler (izquierda) y Runge-Kutta (derecha).




Detalle curva $(x(t), y(t))$




Detalle gráfica $(x(t)$

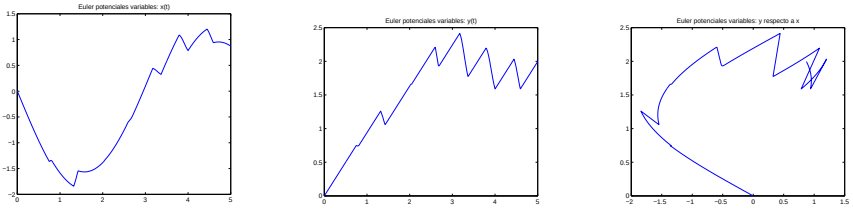



Detalle gráfica $y(t)$

Figura 2.13: Detalles Figuras 2.11, 2.12

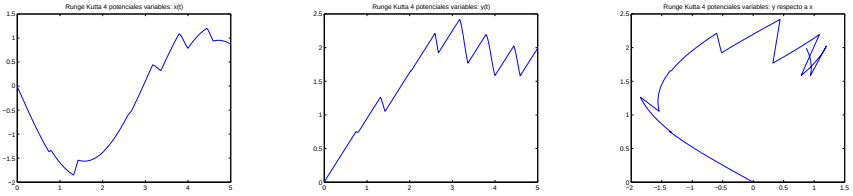
Finalmente, modificamos el nivel de aproximación del problema regularizado tomando $\lambda = 0.01$ en lugar de $\lambda = 0.001$, y mantenemos el número de nodos en las discretizaciones, con lo que parecen atenuarse las oscilaciones.

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.01 |
| b | 5 | x0 | 0 |
| M | 1000 | y0 | 0 |



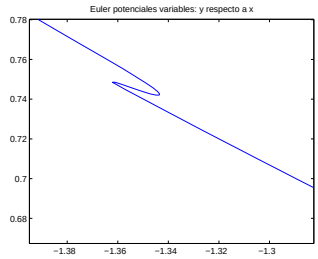

$x(t)$ $y(t)$ $(x(t), y(t))$

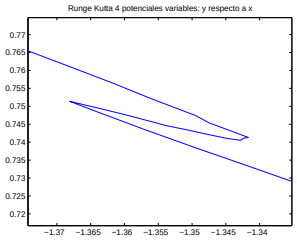
Figura 2.14: Algoritmo de Euler, $\lambda = 0.01$



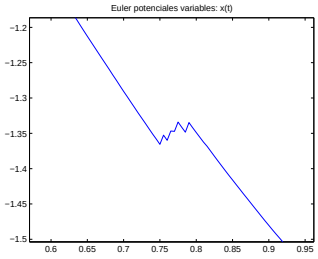

$x(t)$ $y(t)$ $(x(t), y(t))$

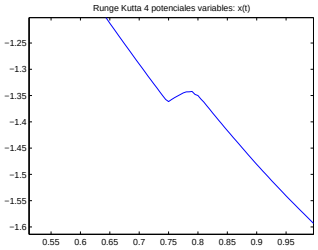
Figura 2.15: Algoritmo de Runge-Kutta, $\lambda = 0.01$



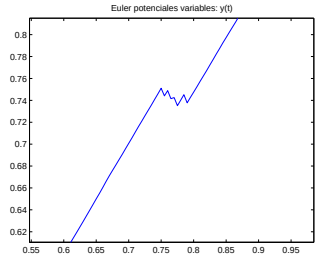



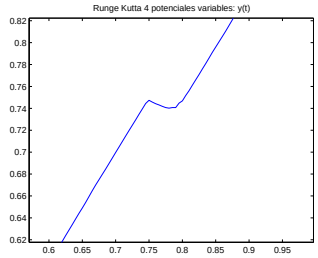

Detalles curva $(x(t), y(t))$




Detalles gráfica $x(t)$




Detalles gráfica $y(t)$

Figura 2.16: Detalles Figuras 2.14, 2.15

## 2.4 Problemas perturbados

La inclusión (1.67) puede perturbarse añadiendo un término fuente dado por un campo vectorial $\mathbf{F} : [0, T[ \times \mathbb{R}^N \longrightarrow \mathbb{R}^N$ continua y de clase $C^1$ en la segunda variable. El problema queda entonces de la forma

$$-\dot{\mathbf{x}}(t) \in \partial\phi(\mathbf{x}(t)) - \mathbf{F}(t, \mathbf{x}(t)) \tag{2.4}$$

Para más detalles de carácter teórico sobre este problema remitimos al capítulo anterior, página 36 y siguientes.

### 2.4.1 Algoritmo de Euler

En el algoritmo no habría que modificar la parte del cálculo de la eresolvente, solamente el tercer paso en que se obtiene el nuevo nodo.

---

**Modificación Euler (problema perturbado)**

---

**3.** Definición del nuevo nodo:

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

---

• **Descripción del código (una variable).** Para este caso se ha definido un nuevo fichero .m denominado `F.m`, conteniendo el término fuente del problema. El programa principal utilizado para problemas de una variable con un término fuente mediante el método de Euler, se denomina `eulerD.m`. La estructura del programa es la misma que para el caso básico, exceptuando la parte donde se actualiza el vector de la variable `x`. En dicha actualización se le añade un nuevo término dependiente del término fuente. Como siempre, una vez acabado el bucle *for*, se presenta la solución en forma de matriz.

• **Descripción del código (dos variables).** La función `eulerD2.m` resuelve problemas de dos variables con el método de Euler para el caso de problemas perturbados. De nuevo cabe comentar que se ha definido en otro fichero .m una función denominada `F2.m`, el cual contendrá la expresión de la función perturbadora de dos variables. Es importante destacar que esta función de dos variables, devolverá un vector de dos componentes.

La estructura del programa es la misma que para el caso básico ya comentado para dos variables, exceptuando la parte donde se actualizan los vectores de las dos variables. Después de calcular el mínimo, se evalúa la función perturbadora utilizando los valores de `x` e `y` de la iteración actual. El resultado se guarda en `F2aux`, cuyo parámetro será un vector de dos elementos. Por tanto a la hora de actualizar el vector de la variable `x` y el de la variable `y`, se debe añadir un nuevo término relacionado con la evaluación de `F2`, tal y como se ha visto

anteriormente. Para ello se escoge la primera coordenada para el vector `X` y la segunda para `Y`.

### 2.4.2 Algoritmo de RK4

Para adaptar el algoritmo RK4 al problema perturbado se añade un nuevo paso en que se calculan los coeficientes asociados a la parte del término fuente y se modifica el tercer paso.

**Modificación RK4 (problema perturbado)**

**2'.** Cálculo de coeficientes asociados a $\mathbf{F}$:

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

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

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

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

**3.** Definición del nuevo nodo:

$$\mathbf{k}_\lambda^{m,j} = \mathbf{k}_{\lambda,1}^{m,j} + 2\mathbf{k}_{\lambda,2}^{m,j} + 2\mathbf{k}_{\lambda,3}^{m,j} + \mathbf{k}_{\lambda,4}^{m,j}$$

$$\mathbf{q}_\lambda^{m,j} = \mathbf{q}_{\lambda,1}^{m,j} + 2\mathbf{q}_{\lambda,2}^{m,j} + 2\mathbf{q}_{\lambda,3}^{m,j} + \mathbf{q}_{\lambda,4}^{m,j}$$

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

• **Descripción del código (una variable).** La función `RK4_D.m` se utiliza para resolver problemas de una variable con término fuente mediante el método de Runge-Kutta de orden cuatro. Se hará uso de los ficheros `calculamin.m` y `yo.m`. Además también se define previamente en otro fichero .m, una función denominada `{F.m`, donde se definirá la función perturbación como se hizo en Euler.

Como se ha mostrado en el esquema anterior, se calculan unos nuevos coeficientes $q_i$ que estarán asociados al término fuente. Cada uno de ellos se irán obteniendo con la evaluación de la función `F` en cada punto correspondiente.

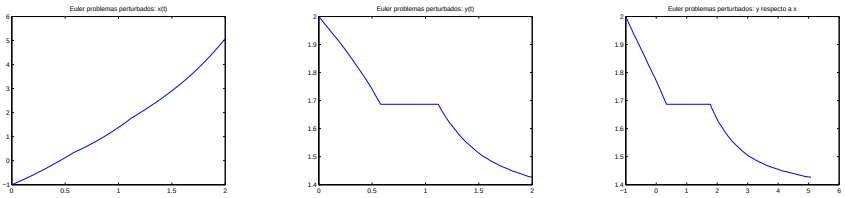
Por tanto la actualización del vector de la variable `x` tendrá sumado un nuevo término dependiente de los nuevos coeficientes.

• **Descripción del código (dos variables).** Para el caso de dos variables tenemos el fichero `RK4_D2.m`, en el cual se define una estructura idéntica al caso de una variable. Simplemente hay que tener en cuenta a la hora de obtener los coeficientes $q_i$, ya que se deben pasar dos argumentos a `feval`. Además a la hora de actualizar los dos vectores asociados a las variables del problema, habrá que tener en cuenta que los coeficientes obtenidos tienen dos componentes, la primera asociada a la variable `x` y la segunda a la variable `y`.

### 2.4.3 Ejemplos de aplicación: Problemas perturbados

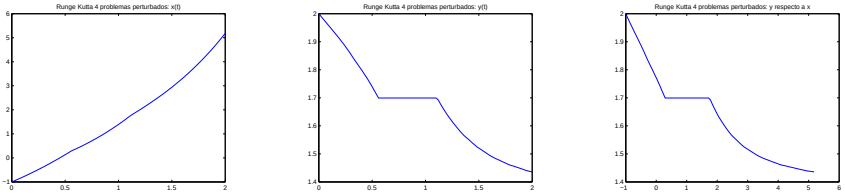
**Ejemplo 2.5** En el primer ejemplo se estudiará el problema asociado a la función distancia al conjunto $C \subset \mathbb{R}^2$ del Ejemplo 2.2, junto con el término fuente dado por el campo vectorial $\mathbf{F}(x, y) = (x + y, 0)$.

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 2 | x0 | -1 |
| M | 100 | y0 | 2 |




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

Figura 2.17: Algoritmo Euler, Ejemplo 2.5



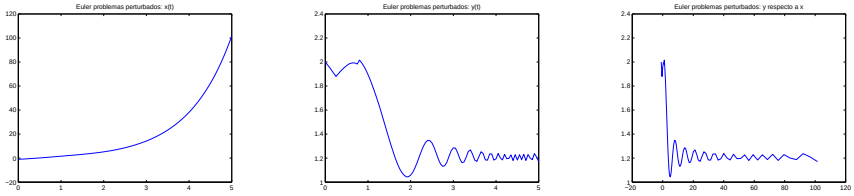

$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.18: Algoritmo RK4, Ejemplo 2.5

**Ejemplo 2.6** Cambiamos ahora el término fuente manteniendo la misma función potencial,

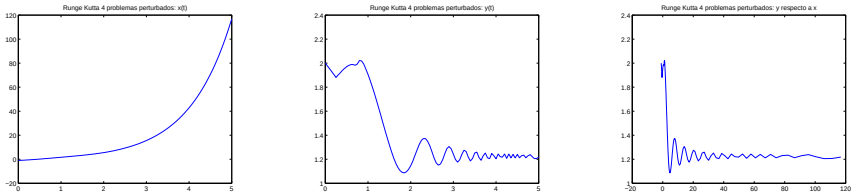
$$\mathbf{G}(x, y) = \begin{cases} (x + y, 0), & x < -0.5 \\ (x + y, \cos(x)), & x \geq -0.5 \end{cases}$$

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 5 | x0 | -1 |
| M | 100 | y0 | 2 |




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

Figura 2.19: Algoritmo Euler, Ejemplo 2.6

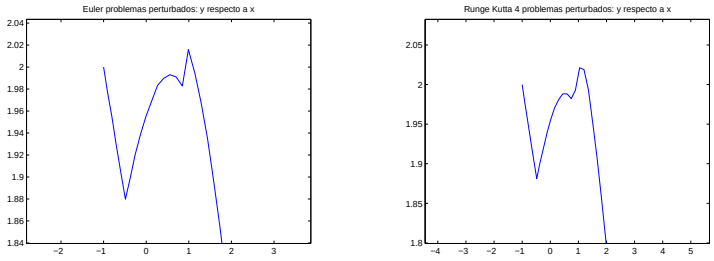



$x(t)$ $y(t)$ $(x(t), y(t))$

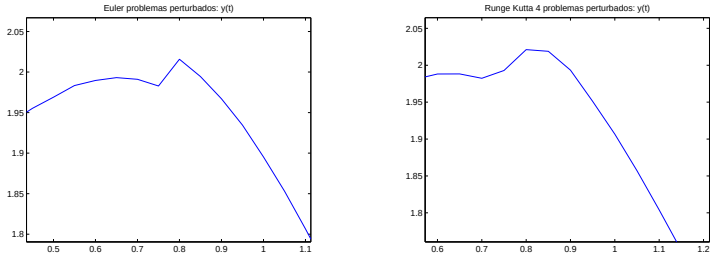
Figura 2.20: Algoritmo RK4, Ejemplo 2.6

Los resultados anteriores muestran la fuerte dependencia del término fuente de las soluciones de problemas del tipo (1.69). En particular se observa que las soluciones de los ejemplos 2.5 y 2.6 son completamente distintas (cabe decir que todos los parámetros son idénticos a excepción del término fuente) y a su vez difieren del caso homogéneo del Ejemplo 2.2.

Por otra parte, en este ejemplo se puede observar en los detalles que incluimos a continuación, que para el método de Runge-Kutta (derecha) las gráficas de la solución son menos abruptas, es decir, se suavizan los picos que se observan en las imágenes obtenidas utilizando el método de Euler (izquierda).




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




Gráfica $y(t)$

Figura 2.21: Detalles Figuras 2.19, 2.20

## 2.5 Potenciales variables y perturbados

Combinando los algoritmos descritos en los apartados anteriores es posible obtener un código para resolver inclusiones diferenciales asociadas a potenciales dependientes del tiempo que además presentan un término fuente (1.69).

Nos limitaremos a describir brevemente los códigos generados en MATLAB y a presentar ejemplos de aplicación.

### 2.5.1 Algoritmo de Euler

• **Descripción del código (una variable).** En este nuevo caso, el fichero `F.m` dependerá de la variable `x` y además del tiempo. Con relación al código principal, denominado `eulerCD.m`, cabe destacar que posee una estructura flexible para tratar casos con potenciales variables y problemas perturbados. Se puede observar la definición del parámetro global `t` usado para evaluar en cada iteración el valor del tiempo, así como la adición del nuevo elemento en la actualización del vector asociado a la variable `x`. Es importante añadir que el valor de `t` se evaluará tanto en la función `phi2.m` tanto como en la perturbación `F.m`.

• **Descripción del código (dos variables).** En este caso se hará uso de la función `F2.m` ya comentada anteriormente, pero con la diferencia de que

ésta posee una dependencia del tiempo. La estructura del programa principal, `eulerCD2.m` es idéntica al programa utilizado para problemas perturbados de dos variables, con la excepción de que hay que añadir el parámetro `t` como global para las diferentes evaluaciones.

### 2.5.2 Algoritmo de RK4

• **Descripción del código (una variable).** En este caso se hace uso de los ficheros `calculamin.m`, `yo.m` y de la función de perturbación `F.m`. Con relación al programa principal, `rk4_CD.m`, éste posee la misma estructura que para el caso de problemas perturbados, con la misma excepción que en el Euler, puesto que se debe definir un parámetro global para el tiempo e ir evaluando para calcular los coeficientes $k_i$ y los $q_i$. Lo único a tener en cuenta es que se debe ir calculando los coeficientes $k_i$ y $q_i$ a la vez, puesto que el valor del parámetro `t` varía según el subíndice que posean los coeficientes, por tanto no se podrán calcular los $q_i$ al final. La actualización de la variable `x` es la misma que para el caso de problemas perturbados.

• **Descripción del código (dos variables).** Se necesita utilizar los ficheros `calculamin2.m`, `yo2.m` y `F2.m`. Simplemente comentar que se trata de la misma estructura que el programa comentado en el punto anterior, con la salvedad de que trabaja con dos variables, y por tanto habría que trabajar en algunas partes del código con vectores.

### 2.5.3 Ejemplos de aplicación: Problemas perturbados y potenciales variables

Implementamos los códigos descritos en dos ejemplos en los que el potencial variable es el del Ejemplo 2.3 y variamos el término fuente.

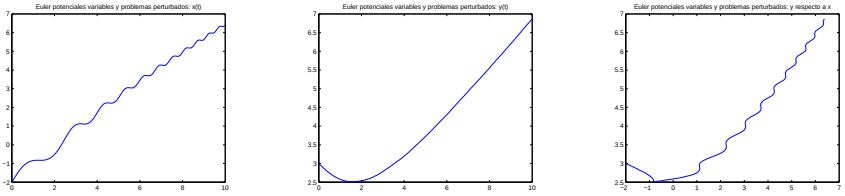
**Ejemplo 2.7** Consideramos en primer lugar el término fuente

$$\mathbf{F}(t, x, y) = (\cos(ty), 0)$$

y mantenemos el resto de parámetros del Ejemplo 2.3.

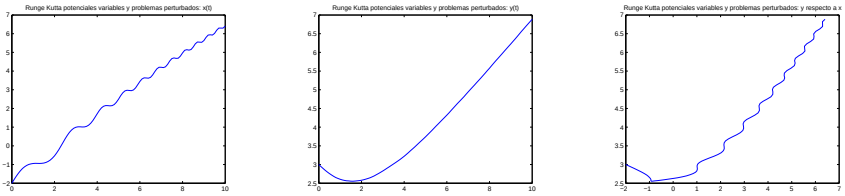
| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| `a` | 0 | `lambda` | `0.001` |
| `b` | 10 | `x0` | `-2` |
| `M` | 100 | `y0` | 3 |

Comparando las gráficas obtenidas con las del Ejemplo 2.3 se observa que son similares, aunque en este caso aparecen oscilaciones debidas a la perturbación provocada por el término fuente. Además, dado que el segundo término del campo $\mathbf{F}$ es idénticamente nulo, la segunda componente de la solución no varía con respecto a la del Ejemplo 2.3.




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

Figura 2.22: Algoritmo Euler, Ejemplo 2.7




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

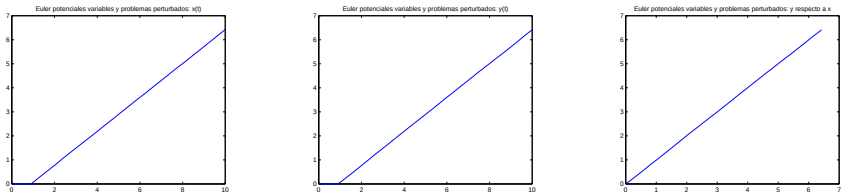
Figura 2.23: Algoritmo RK4, Ejemplo 2.7

**Ejemplo 2.8** En la segunda simulación se usa como término fuente el campo vectorial

$$\mathbf{G}(t,x,y) = \begin{cases} (0,0), & \text{si } d_{C(t)}(x,y) \geq 0.1 \\ (y,x), & \text{en otro caso} \end{cases}$$

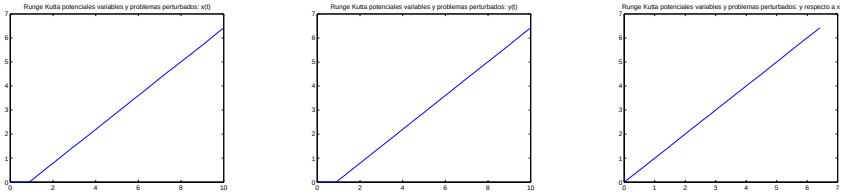
con los siguientes parámetros

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 10 | x0 | 0 |
| M | 100 | y0 | 0 |




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

Figura 2.24: Algoritmo Euler

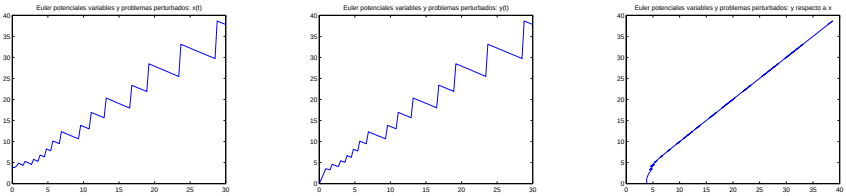



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.25: Algoritmo RK4

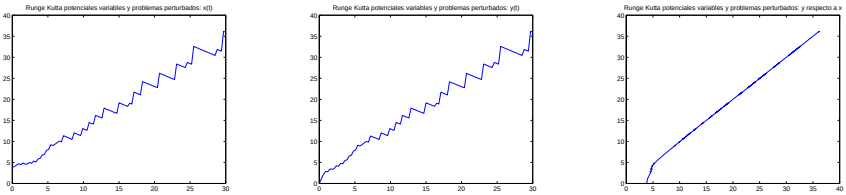
Para el mismo problema realizamos una nueva simulación modificando la longitud del intervalo y las condiciones iniciales.

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| `a` | `0` | `lambda` | `0.001` |
| `b` | `30` | `x0` | `1+sqrt(8)` |
| `M` | `100` | `y0` | `0` |




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

Figura 2.26: Algoritmo Euler




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

Figura 2.27: Algoritmo RK4

De nuevo se observa la aparición de oscilaciones, más amortiguadas cuando se utiliza el algoritmo de Runge-Kutta.

## 2.6 Problemas bipotenciales

Otra clase de problemas asociados a gradientes son los denominados de tipo bipotencial, descritos en el capítulo anterior, que tienen la forma general

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

donde $\phi, \varphi : \mathbb{R}^N \longrightarrow \mathbb{R}$ son funciones convexas.

### 2.6.1 Algoritmo de Euler

En este caso, para obtener el problema regularizado, habrá que añadir un nuevo paso al algoritmo de Euler para calcular resolvente de Moreau asociada a la otra función $\varphi$. También se modificará la actualización de los nodos en la discretización.

---

**Modificación Euler (problemas bipotenciales)**

---

**2'.** Cálculo de la resolvente de Moreau:

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

**3.** Definición del nuevo nodo:

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

---

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

Cabe mencionar asimismo que se ha considerado el caso en que $\beta$ es un función que depende exclusivamente del tiempo, dado que es el único problema de este tipo que ha sido tratado en la bibliografía, sin embargo, desde el punto de vista computacional, sería factible considerar situaciones más complicadas en las que $\beta$ pueda depender también del estado $\mathbf{x}$.

• **Descripción del código (una variable).** En este caso será necesario definir dos ficheros .m para las funciones que se van a utilizar. Dichos ficheros se denominan `f.m` y `g.m`. En cada una de ellas se han definido como parámetro globales `cx`, `cy`, `t` y el nivel de aproximación asociado a cada potencial: `lambda1` para `f` y `lambda2` para la función `g`.

El programa principal de este apartado se denomina `eulerE.m` y sus argumentos de entrada son los mismos que los vistos hasta ahora.

Dentro del bucle *for*, la primera parte relativa a la asignación del valor a `cx` no varía. Tras asignar al parámetro `t` el valor que posee el vector `T` en la iteración en la que se encuentre el bucle, seguidamente se define la función denominada en nuestro programa como `epsilon`, la cual dependerá del tiempo. Un cambio significativo es que se produce dos cálculos del minímo, uno para cada función, siempre empezando la búsqueda en el mismo punto. Otra variación importante es la actualización del valor de la variable `x`, ya que ésta depende de los dos valores de los niveles de aproximación así como de los dos minímos obtenidos y de $\varepsilon(t)$, quedando de la siguiente manera en MATLAB:

```
X(j+1) = X(j) - (h/lambda1)*(X(j)-minimof)
              - epsilon*(h/lambda2)*(X(j)-minimog)
```

• **Descripción del código (dos variables).** El programa principal para este apartado se denomina `eulerE2.m` y posee la misma estructura que `eulerE.m` con la salvedad de que se trabaja con dos variables. Simplemente cabe destacar que el punto para comenzar con el cálculo de los minímos debe ser un vector de dos componentes, y que la actualización de cada una de las variables dependerá de las primeras coordenadas de los minímos si se trata de la variable `x` y de las segundas si se trata de la variable `y`.

### 2.6.2 Algoritmo de RK4

La modficación del algoritmo RK4 es algo más complicada, ya que aparte de introducir un nuevo proceso de minimización para calcular la resolvente del segundo potencial $\varphi$ hay que calcular cuatro nuevos coeficientes que afectan a la definición de cada nodo a partir del anterior, asumiendo niveles de aproximación diferentes para la regularización de Yosida de cada subdiferencial.

---

**Modificación RK4 (problemas bipotenciales)**

---

**2.** Cálculo de los coeficientes asociados a $\phi$:

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

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

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

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

donde

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

**2'.** Cálculo de los coeficientes asociados a $\varphi$:

$$\mathbf{q}_{\mu,1}^{m,j} = \mathbf{x}_{\lambda,\mu}^{m,j} - I_\mu(\mathbf{x}_{\lambda,\mu}^{m,j})$$

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

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

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

donde

$$I_\mu(\mathbf{z}) = \underset{\mathbf{y}\in\mathbb{R}^N}{\operatorname{argmin}} \left( \varphi(\mathbf{y}) + \frac{1}{2\mu} |\mathbf{y}-\mathbf{z}|^2 \right)$$

**3.** Definición del nuevo nodo:

$$\mathbf{k}_\lambda^{m,j} = \mathbf{k}_{\lambda,1}^{m,j} + 2\mathbf{k}_{\lambda,2}^{m,j} + 2\mathbf{k}_{\lambda,3}^{m,j} + \mathbf{k}_{\lambda,4}^{m,j}$$

$$\mathbf{q}_\mu^{m,j} = \mathbf{q}_{\mu,1}^{m,j} + 2\mathbf{q}_{\mu,2}^{m,j} + 2\mathbf{q}_{\mu,3}^{m,j} + \mathbf{q}_{\mu,4}^{m,j}$$

$$\mathbf{x}_{\lambda,\mu}^{m,j+1} = \mathbf{x}_{\lambda,\mu}^{m,j} - \frac{h_m}{6}\left( \frac{\mathbf{k}_\lambda^{m,j}}{\lambda} + \beta(t^{m,j})\frac{\mathbf{q}_\mu^{m,j}}{\mu} \right)$$

---

• **Descripción del código (una variable).** El código `rk4_E.m` resuelve problemas de tipo bipotencial usando el método de Runge-Kutta de cuarto orden para aproximar el problema regularizado. Con el fin de agilizar la implementación y ahorrar código se usarán diversos ficheros `.m`. Comenzaremos describiendo y comentando la funcionalidad de cada uno de los archivos utilizados y finalmente el principal denominado `rk4_E.m`.

♢ FUNCIÓN `yof.m`:

Define la resolvente para la función `f.m` en el caso de una variable, ya que posee un parámetro global denominado `minimof`, el cual hace referencia al mínimo calculado en la función `f`.

♢ FUNCIÓN `yog.m`:

Define la resolvente una variable para la función `g.m` ya que posee un parámetro global denominado `minimog`, el cual hace referencia al mínimo calculado en la función `g`.

Los ficheros `yof.m` y `yog.m` son idénticos en cuanto a estructura y funcionalidad a `yo.m`.

♢ FUNCIÓN `calculaminf.m`:

Calcula el mínimo de la función `f.m`, pasándole como argumento el punto donde se comenzará a buscarlo.

♢ FUNCIÓN `calculaming.m`:

Calcula el mínimo de la función `g.m`.

Ocurre lo mismo para estas dos últimas funciones, ya que son idénticas a `calculamin.m`, con la diferencia de que se particulariza para cada función.

♢ FUNCIÓN `rk4_E.m`:

Se han definido siete nuevos parámetros globales: los asociados a los dos niveles de aproximación (`lambda1` y `lambda2`), los asociados a los mínimos que se van calculando en el algoritmo (`minimof` y `minimog`), los relativos al punto de comienzo de búsqueda del mínimo en cada caso y evaluación de la resolvente para cada función `f` y `g` (`puntoxf` y `puntoxg`) y el parámetro `t`.

Al igual que el código de Euler de una variable, se establecen las condiciones iniciales según una estructura de decisión *if/else*, se asigna a `t` el valor de `T(j)` y se define `epsilon`. La parte donde existen cambios es la relativa al cálculo de los coeficientes $k_i$ y $q_i$. Inicialmente se calculan los $k_i$, que son los referentes a la función `f.m`. Para ello se hace uso del parámetro `puntoxf` para ir definiendo el punto a evaluar y de las funciones previamente definidas: `calculaminf` y `yof`. A continuación se calculan los $q_i$, los cuales son los asociados a la función `g.m`. Finalmente se realiza la actualización del vector de la variable `x`, de forma que queda de la siguiente manera:

```
X(j+1)= X(j) + (h/6)*(k1+2*k2+2*k3+k4)
             - epsilon*(h/6)*(q1+2*q2+2*q3+q4)
```

• **Descripción del código (dos variables).** La función `rk4_E2.m` resuelve problemas de tipo bipotencial con dos variables con el método de Runge-Kutta de orden cuatro. Haremos uso de las funciones ya descritas anteriormente: `calculaminf.m` y `calculaming.m`. Además se definirán nuevas funciones a utilizar para evitar repeticiones de código en el programa.

♢ FUNCIÓN `yo2f.m`:

Define la resolvente de la función `f.m` en el caso bidimensional. Es interesante incluir su estructura debido a que se define sin incluir el producto por $-1/\lambda$ tal y como se hacía en `yo2.m`.

```
yo_1 = (x(1)-minimof(1));
yo_2 = (x(2)-minimof(2));
p = [yo_1,yo_2];
```

♢ FUNCIÓN `yo2g.m`:

Análogo a la función anterior para `g.m`.

♢ FUNCIÓN `rk4_E2.m`:

Para dos variables, se definen además de los definidos para una variable, los siguientes parámetros globales: los asociados a la segunda coordenada del punto de comienzo de búsqueda del mínimo en cada caso y evaluación de la resolvente para cada función `f` y `g` (`puntoyf` y `puntoyg`).

Con relación al cálculo de los coeficientes $k_i$ y $q_i$, se realiza del mismo modo que para una variable, pero teniendo en cuenta que ahora trabajamos con dos variables. Por lo tanto, tras indicar el valor que poseerán `puntoxf`, `puntoyf`, `puntoxg` y `puntoyg`, se crearán vectores de dos componentes que constarán de sus respectivas coordenadas para cada función. De esta manera se podrá pasar como argumento y así calcular los mínimos y finalmente los coeficientes $k_i$ y $q_i$.

Una vez calculados todos los coeficientes, en cada iteración se irán actualizando los vectores asociados a las variables `x` e `y`. La expresión será la misma que la comentada para una variable, pero teniendo en cuenta que los coeficientes constarán de dos componentes, por tanto, para cada variable se seleccionará la coordenada pertinente: para `x` la primera y para `y` la segunda.

### 2.6.3 Ejemplos de aplicación: Problemas bipotenciales

Como en todos los casos, utilizamos una serie de problemas concretos para validar los códigos.

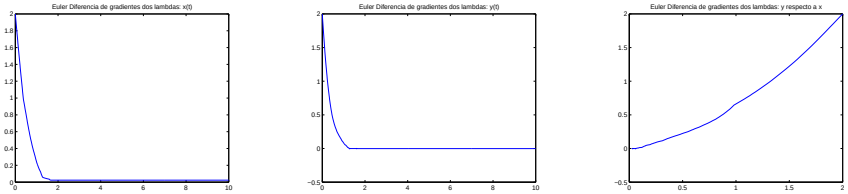
**Ejemplo 2.9** Sean $I = [a_1, b_1]$, $J = [a_2, b_2] \subset \mathbb{R}$, dos intervalos compactos. Se consideran los potenciales convexos

$$\phi(x, y) = d_I(x) + d_J(y) + x^2 \tag{2.5}$$

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

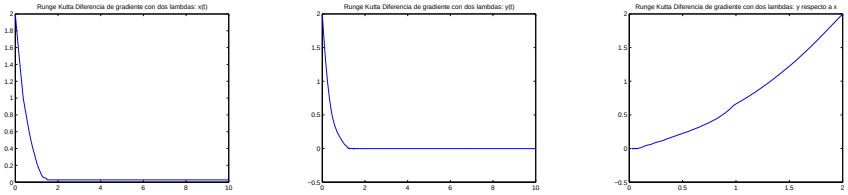
Se ejecuta los códigos MATLAB para aproximar la solución del (1.70) para estas funciones concretas y con los valores de los parámetros indicados en la tabla.

| 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 2.28: Euler, $\lambda = \mu = 0.001$

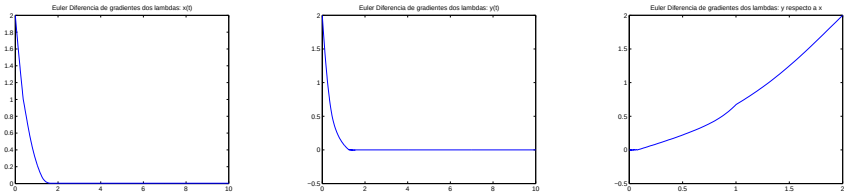



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.29: Runge-Kutta, $\lambda = \mu = 0.001$

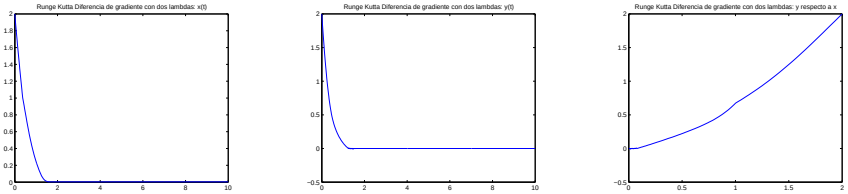
Del artículo de Attouch y Czarnecki [1] sabemos que la solución de este problema, independientemente del punto inicial, converge a $(0, 0)$ cuando $t \to +\infty$. En nuestras simulaciones, para $t = 10$ se obtiene el valor $(0.0236, -0.0016)$ para el algoritmo de Euler y $(0.0269, -0.0017)$ para el de Runge-Kutta.

Con objeto de valorar la sensibilidad de los algoritmos respecto del nivel de aproximación de las regularizaciones de Yosida de cada sundiferencial, repetimos las simulaciones anteriores manteniendo el valor de $\lambda$ (`lambda1=0.001`) y modificando el de $\mu$ (`lambda2=0.01`). Se observa que aunque la forma de las gráficas es similar, el valor final varía notablemente respecto de la simulación anterior: $(0.0039, -0.0003)$ para Euler y $(0.0039, -0.0001)$ para Runge-Kutta.




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

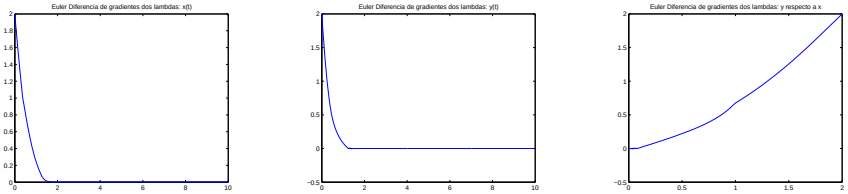
Figura 2.30: Euler, $\lambda = 0.001,\ \mu = 0.01$




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

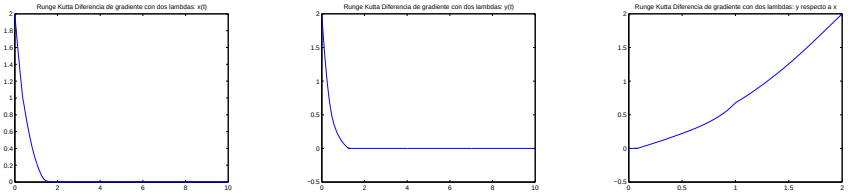
Figura 2.31: Runge-Kutta, $\lambda = 0.001,\ \mu = 0.01$

Finalmente, tomando los valores $\lambda = 0.0014$, $\mu = 0.013$, para $t = 10$ el algoritmo de Euler proporciona el valor $(0.0018, 0)$ y el de Runge-Kutta $(0.0019, 0)$.




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

Figura 2.32: Euler, $\lambda = 0.014$, $\mu = 0.013$

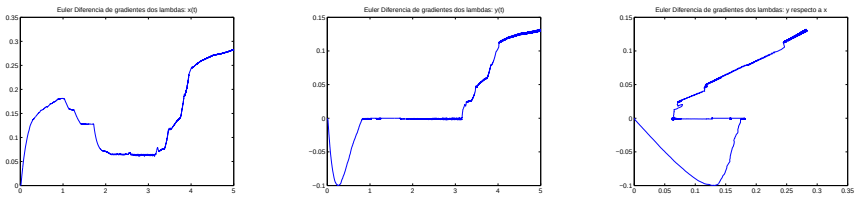



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.33: Runge-Kutta, $\lambda = 0.014$, $\mu = 0.013$

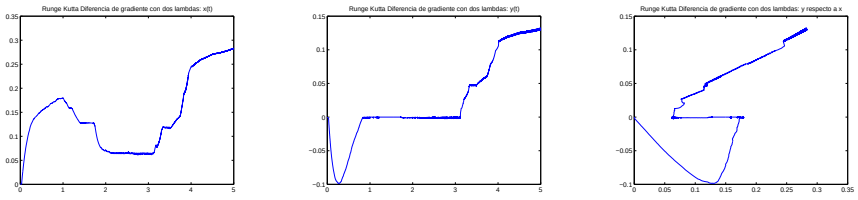
**Ejemplo 2.10** Vamos a considerar ahora el mismo problema que en el ejemplo anterior, cambiando los extremos de los intervalos $I$, $J$. Los valores de los distintos parámetros se recopilan en la siguiente tabla.

| Parámetro | Valor | Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|---|---|
| a | 0 | x0 | 0 | a2 | -1 |
| b | 5 | y0 | 0 | b2 | -0.5 |
| M | 5000 | a1 | 0.5 | a3 | 1 |
| lambda1 | 0.001 | b1 | 1 | b3 | 2 |
| lambda2 | 0.001 | | | | |




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

Figura 2.34: Euler, Ejemplo 2.10, $\lambda = \mu = 0.001$

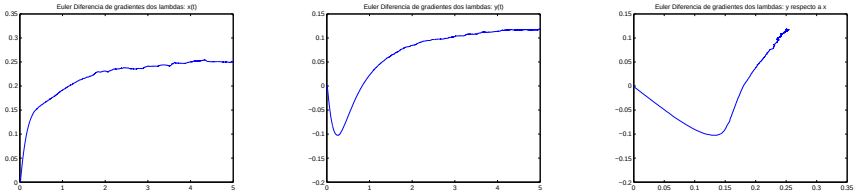



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.35: Runge-Kutta, Ejemplo 2.10, $\lambda = \mu = 0.001$

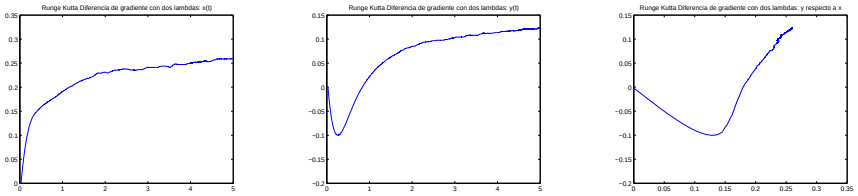
De la referencia [1], Eq. (23), sabemos que cuando $t \to +\infty$ la solución del problema debe converger a $(0.25, 0.125)$. Nuestras simulaciones proporcionan para $t = 5$ los valores $(0.2825, 0.1317)$ (Euler) y $(0.2825, 0.1317)$ (Runge-Kutta).

Las siguientes gráficas han sido generadas para los mismos datos cambiando el nivel de aproximación de la regularización de Yosida de la segunda subdiferencial, $\mu = 0.01$.




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

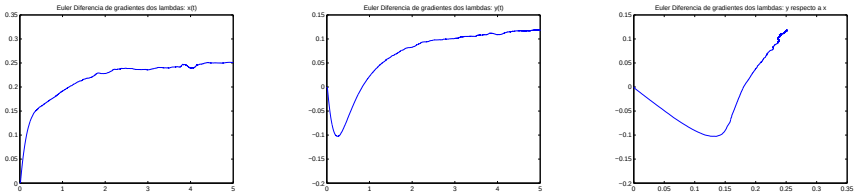
Figura 2.36: Euler, Ejemplo 2.10, $\lambda = 0.001$, $\mu = 0.01$




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

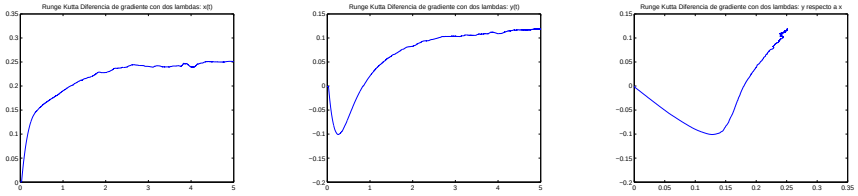
Figura 2.37: Runge-Kutta, Ejemplo 2.10, $\lambda = 0.001$, $\mu = 0.01$

En este caso para $t = 5$ se obtienen los siguientes valores para la solución: $(0.2506, 0.1187)$ usando el método de Euler y $(0.2600, 0.1235)$ para Runge-Kutta. Finalmente, si $\mu = 0.011$, los valores obtenidos en $t = 5$ son $(0.2509, 0.118)$ para Euler y $(0.2509, 0.118)$ para Runge-Kutta.




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

Figura 2.38: Euler, Ejemplo 2.10, $\lambda = 0.001$, $\mu = 0.011$

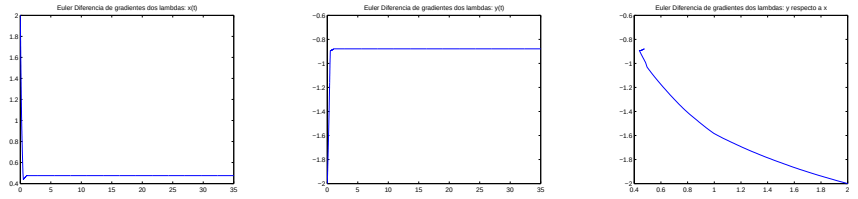



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 2.39: Runge-Kutta, Ejemplo 2.10, $\lambda = 0.001$, $\mu = 0.011$

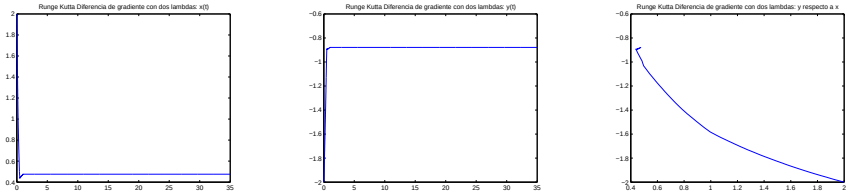
**Ejemplo 2.11** Para finalizar se consideran las mismas funciones $\phi$ y $\beta$ que en el Ejemplo 2.9 y se toma $\varphi$ como la función distancia a la bola unidad $\mathcal{B} = \left\{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\right\}$. El valor de los distintos parámetros viene dado por la tabla.

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | x0 | 2 |
| b | 35 | y0 | -2 |
| M | 10000 | a1 | 0.5 |
| lambda1 | 0.001 | b1 | 1 |
| lambda2 | 0.001 | a2 | -1 |
| | | b2 | -0.5 |




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

Figura 2.40: Euler, Ejemplo 2.11




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

Figura 2.41: Runge-Kutta, Ejemplo 2.11

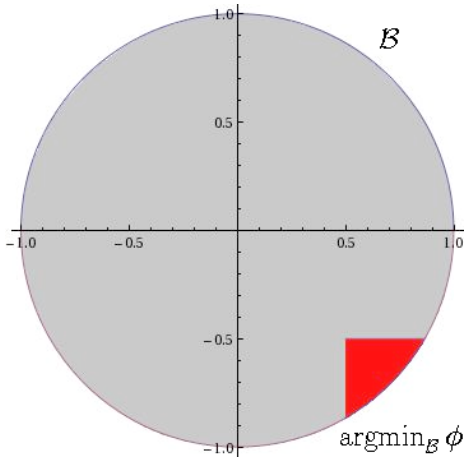
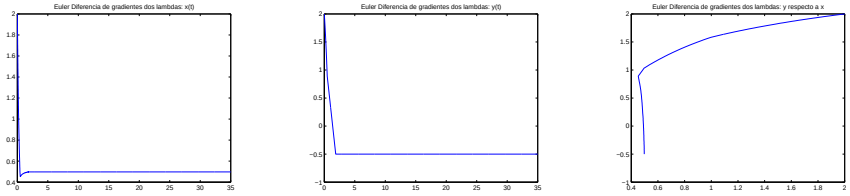



Figura 2.42: Conjunto $\operatorname{argmin}_{\mathcal{B}} \phi$

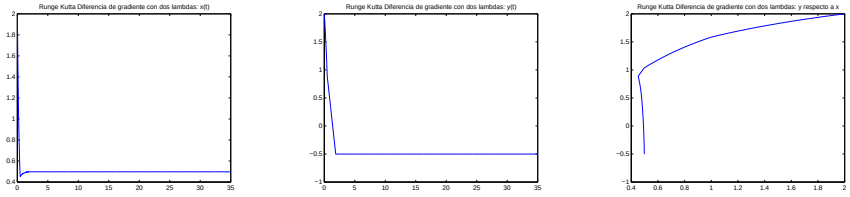
Del Th. 3.1 en [1] (ver Teorema 1.18), se sigue que cuando $t \to +\infty$, la solución del problema converge a un punto en $\operatorname{argmin}_{\mathcal{B}} \phi = \mathcal{B} \cap [0.5, 1] \times [-1, -0.5]$ (en rojo en la Figura 2.42). Las simulaciones están de acuerdo con este resultado ya que para $t = 35$, obtenemos los valores $(0.4757, -0.8795)$ (Euler) y $(0.4758, -0.8789)$ (Runge-Kutta), que nos permiten conjeturar que la solución converge asintóticamente a al punto $(0.5, -0.866025) \in \operatorname{argmin}_{\mathcal{B}} \phi$.

Finalmente simulamos la solución a partir del punto inicial $(2, 0)$, en lugar del $(0, 0)$, tomando todos los parámetros iguales a excepción de $\mu = 0.01$.




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

Figura 2.43: Euler, Ejemplo 2.11, $x_0 = 2$




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

Figura 2.44: Runge-Kutta, Ejemplo 2.11, $x_0 = 2$

Respecto al comportamiento asintótico, para $t = 35$ los algoritmos porporcionan los valores $(0.4975, -0.5003)$ (Euler) y $(0.4975, -0.5008)$ (Runge-Kutta), lo que nos permite conjeturar que la solución tenderá hacia $(0.5, -0.5) \in \operatorname{argmin}_{\mathcal{B}} \phi$.

# Capítulo 3

# Experimentos numéricos

En el presente capítulo se presentan diversos experimentos y simulaciones numéricas que permiten por un lado comprobar el adecuado funcionamiento de los algoritmos descritos en el Capítulo 2 y sus implementaciones en MATLAB y por otro obtener información relevante sobre algunos problemas interesantes.

Se consideran fundamentalmente inclusiones de tipo subdiferencial cuyos potenciales son funciones distancia a conjuntos convexos y problemas bipotenciales. Los códigos asociados se encuentran en los Anexos SO.

## 3.1 Potenciales del tipo $d_C(\cdot)$

Al considerar problemas del tipo,

$$-\dot{\mathbf{x}}(t) \in \partial d_C(\mathbf{x}(t)) \tag{3.1}$$

donde $C \subset \mathbb{R}^N$ es un conjunto convexo con $N \geq 2$, que puede ser constante o variable a lo largo del tiempo, se presenta una dificultad adicional a la hora de implementar el algoritmo de Moreau-Yosida para aproximar la solución, independiente del método numérico que se use para resolver los problemas regularizados. En efecto, para calcular la resolvente es necesario resolver problemas del tipo

$$\underset{\mathbf{y}\in\mathbb{R}^N}{\operatorname{argmin}} \left( d_C(\mathbf{y}) + \frac{1}{2\lambda}|\mathbf{y}-\mathbf{x}|^2 \right) \tag{3.2}$$

lo que exige conocer expícitamente el valor de la función distancia para poder usar el comando `fminsearch`. Pero esto solamente es posible en algunos casos concretos, por ejemplo si $C$ es un círculo (Ejemplo 2.2).

### 3.1.1 Algoritmos modificados

La dificultad se solventa modificando ligeramente el algoritmo para el cálculo de la resolvente, teniendo en cuenta el resultado siguiente.

**Lema 3.1** *Sea $C \subset \mathbb{R}^N$ un conjunto convexo y cerrado no vacío. Si para cada $\lambda > 0$, $\mathbf{x} \in \mathbb{R}^N$, $J_\lambda(\mathbf{x})$ es la solución de (3.2), se tiene que $(J_\lambda(\mathbf{x}), \mathrm{proj}(J_\lambda(\mathbf{x}); C))$ es la única solución de*

$$\operatorname*{argmin}_{\mathbf{y}\in\mathbb{R}^N, \mathbf{z}\in C} \left( |\mathbf{y}-\mathbf{z}| + \frac{1}{2\lambda}|\mathbf{y}-\mathbf{x}|^2 \right), \tag{3.3}$$

*donde* $\mathrm{proj}(\cdot; C)$ *denota la proyección ortogonal sobre* $C$.

En efecto, la función $(\mathbf{y}, \mathbf{z}) \rightsquigarrow |\mathbf{y}-\mathbf{z}| + \frac{1}{2\lambda}|\mathbf{y}-\mathbf{x}|^2$ es estrictamente convexa[(1)], por lo que el problema (3.3) tendrá a lo sumo una única solución (Proposición 1.8). Pero, al ser $J_\lambda(\mathbf{x})$ solución de (3.2), para cada $\mathbf{y} \in \mathbb{R}^N$

$$\begin{aligned} &|J_\lambda(\mathbf{x}) - \mathrm{proj}(J_\lambda(\mathbf{x}); C)|^2 + \frac{1}{2\lambda}|J_\lambda(\mathbf{x}) - \mathbf{x}|^2 \\ &\qquad \leq d_C(\mathbf{y}) + \frac{1}{2\lambda}|\mathbf{y}-\mathbf{x}|^2 \leq |\mathbf{y}-\mathbf{z}| + \frac{1}{2\lambda}|\mathbf{y}-\mathbf{x}|^2 \end{aligned}$$

para cada $\mathbf{z} \in C$ por definición de distancia, lo que nos da la identidad buscada.

Lo que nos dice el lema anterior es que es posible calcular la resolvente asociada a una función distancia sin conocer su expresión explícita, basta con calcular un mínimo en $\mathbb{R}^N \times \mathbb{R}^N$. En el caso del algoritmo de Euler se modifica el segundo paso de la forma:

---

**Algoritmo de Euler para la función distancia**

---

**2'.** Cálculo de la resolvente:

$$\mathbf{y}_\lambda^{m,j} = \mathrm{proj}\left( \operatorname*{argmin}_{\mathbf{y}\in\mathbb{R}^N,\, \mathbf{z}\in C} \left( |\mathbf{y}-\mathbf{z}| + \frac{1}{2\lambda}\left|\mathbf{y}-\mathbf{x}_\lambda^{m,j}\right|^2 \right); \mathbb{R}^N \right)$$

$\mathrm{proj}\left(\cdot; \mathbb{R}^N\right)$ selecciona las $N$ primeras componentes.

---

Análogamente, en el algoritmo de Runge-Kutta se hace la siguiente modificación:

---

[(1)] Obviamente es convexa. Además, la función $g_\lambda(\mathbf{y}) = \frac{1}{2\lambda}|\mathbf{y}-\mathbf{x}|^2$ es de clase $C^1$, con $\nabla g_\lambda(\mathbf{y}) = \frac{1}{\lambda}(\mathbf{y}-\mathbf{x})$. Teniendo en cuenta que $\langle \nabla g_\lambda(\mathbf{y}) - \nabla g_\lambda(\mathbf{z}), \mathbf{y}-\mathbf{z}\rangle = \frac{1}{\lambda}|\mathbf{y}-\mathbf{z}|^2$, de la Proposición 1.9 se deduce que $g_\lambda$ es estrictamente convexa, luego la función completa también lo es.

**Algoritmo de RK4 para la función distancia**

**2'.** Cálculo de los coeficientes:

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

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

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

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

donde

$$J_\lambda(\mathbf{u}) = \text{proj}\left(\underset{\mathbf{y}\in\mathbb{R}^N,\,\mathbf{z}\in C}{\text{argmin}}\left(|\mathbf{y}-\mathbf{z}| + \frac{1}{2\lambda}|\mathbf{y}-\mathbf{u}|^2\right); \mathbb{R}^N\right)$$

### 3.1.2 Modificaciones en los códigos

Para calcular el nuevo mínimo en el segundo paso se utiliza el código libre *SolvOpt*, que permite determinar el mínimo (con o sin restricciones) de una función de varias variables (dos o más), no necesariamente diferenciable mediante el algoritmo de Shor (ver [12])

Para usar el código *SolvOpt*, cuyo fichero principal del optimizador se denomina `solvopt.m`, dado que buscamos un mínimo con restricciones se deben definir un fichero donde se encuentre la función objetivo, que en nuestro caso viene dada por la expresión en **2'.**, y otro fichero donde se fijarán las restricciones. A continuación se presenta la cabecera de la función donde se definen tanto los argumentos de entrada como los de salida:

```
function [x,f,options]=solvopt(x,fun,grad,options,func,gradc)}
```

donde los argumentos de entrada que utilizaremos son los siguientes:

- `x`: es el vector de tamaño `n` con las coordenadas del punto de inicio.
- `fun`: es el nombre del fichero m donde se tendrá definida la función objetivo.
- `func`: es el nombre del fichero .m donde se establecen las restricciones de nuestro problema.

En relación a los argumentos de salida que serán de nuestro interés, se tiene:

- `x`: es el vector de los valores asociados al mínimo encontrado.

Para más detalles remitimos a [12].

Para unificar notación, en todos los casos llamaremos `funobjetivo.m` al fichero que contiene la función objetivo, con la única diferencia de un cambio de nomenclatura enncada método utilizado, y `funcMr.m` al fichero donde se describen las restricciones. En cuanto a los resultados, solo nos interesará el valor obtenido del mínimo, en realidad sus $N$ primeras coordenadas (en nuestras simulaciones $N = 2$).

En general se han sido necesarias únicamente dos modificaciones para adaptar el código:

- La definición de dos nuevos parámetros globales denominados de forma general `g1` y `g2`, los cuales formarán parte del vector creado como argumento de entrada `x`. Dicho vector de tamaño `n` posee las coordenadas del punto de inicio, de forma que las dos primeras serán las condiciones iniciales de las variables `x` e `y`, y las dos restantes serán los parámetros globales comentados, los cuales dependerán del tipo de restricción que establezcamos.Por ejemplo, en el caso de un círculo dichos parámetros hacen referencia a las coordenadas del centro del mismo.

- Cambio de la línea donde se calcula el mínimo, colocando en su lugar el siguiente código con *SolvOpt*:

```
vector=[X(1),Y(1),centrox,centroy];
[mini,f] = solvopt(vector,'funobjetivo',[],[],'funcMr',[]);
minimo = [mini(:,1),mini(:,2)];
```

  donde `vector` define el punto de inicio de búsqueda. Se puede observar que a la hora de llamar a `solvopt.m` se usan como argumentos de entrada el vector definido, el nombre del fichero de la función objetivo y el del fichero donde se encuentran las restricciones, en los demás se han colocado dos corchetes vacíos. Esta línea nos devolverá los parámetros `mini` y `f`, donde sólo nos interesa el valor de `mini`, el cual se trata de un vector de tamaño 4. De dicho vector escogeremos las dos primeras coordenadas que hacen referencia a las dos variables que estamos estudiando (`x` e `y`), obteniendo así el valor de las coordenadas del mínimo que corresponde con el parámetro `minimo` utilizado en las versiones iniciales.

En el caso particular del método de Runge-Kutta en se deben tener en cuenta ciertos aspectos:

- La estructura del fichero utilizado para el cálculo del mínimo (*calculaminSO2.m*) cambia de optimizador y también a la hora de devolver su resultado, debido a que solo nos interesa las dos primeras coordenadas del mínimo obtenido con SolvOpt.

- Se debe definir en cada parte del algoritmo donde se calculan los coeficientes $k_i$ los valores de los parámetros globales utilizados en `vector`. Debido a que éstos pueden variar con el tiempo, se debe ir repitiendo después de la asignación del valor del parámetro `t`.

### 3.1.3 Simulaciones

**Ejemplo 3.1** Consideremos la siguiente familia de conjuntos móviles

$$C(t) = \left\{(x, y) \in \mathbb{R}^2 : (x - \cos(t))^2 + (y - \text{sen}(t))^2 \leq 0.25\right\}$$

representada en la Figura 3.1

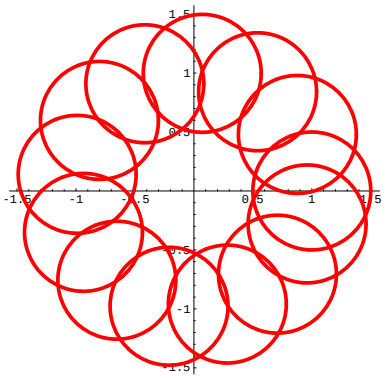
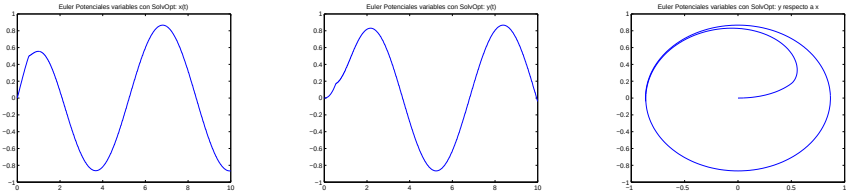



Figura 3.1: Círculos centro móvil

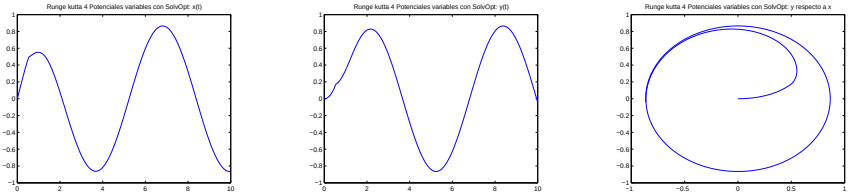
Usando los códigos modificados se resuelve el problema (3.1) de acuerdo con los siguientes parámetros. Los resultados se presentan en las Figuras 3.2 y 3.3.

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| `a` | `0` | `x0` | `0` |
| `b` | `10` | `y0` | `0` |
| `M` | `1000` | `g1` | `cos(t)` |
| `lambda` | `0.001` | `g2` | `sin(t)` |



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

Figura 3.2: Círculos centro móvil (Euler)



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

Figura 3.3: Círculos centro móvil (RK4)

**Ejemplo 3.2** Podemos asimismo considerar el caso en que también varían los radios de los círculos, por ejemplo,

$$C(t) = \left\{(x, y) \in \mathbb{R}^2 : (x - \cos(t))^2 + (y - \operatorname{sen}(t))^2 \leq (0.5 + 0.25\cos(2t))\right\}$$

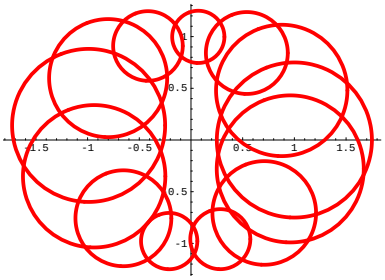
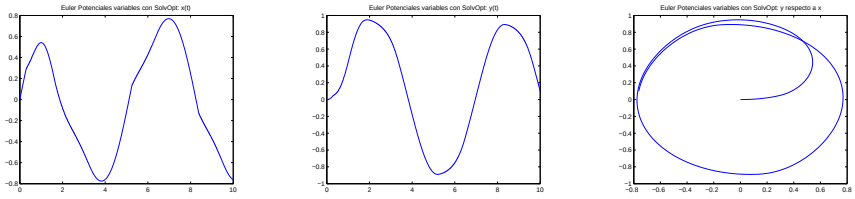



Figura 3.4: Círculos centro y radio móviles

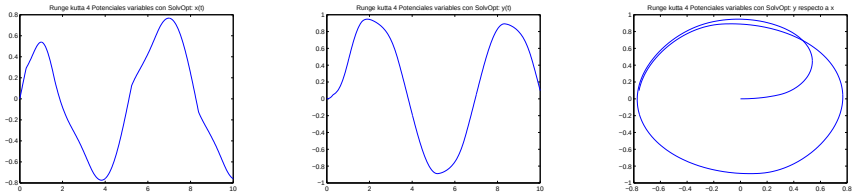
Se observa que para los mismos valores de los parámetros se obtienen resultados similares (Figuras 3.5 y 3.6).

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | x0 | 0 |
| b | 10 | y0 | 0 |
| M | 1000 | g1 | cos(t) |
| lambda | 0.001 | g2 | sen(t) |
| | | radio | 0.5+0.25*cos(2*t) |




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

Figura 3.5: Círculos centro y radio móviles (Euler)




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

Figura 3.6: Círculos centro y radio móviles (RK4)

Realizamos ahora diferentes simulaciones cambiando las condiciones iniciales para ver cómo varían las soluciones. Dado que prácticamente no se aprecian diferencias, presentamos únicamente los resultados obtenidos mediante el código Runge-Kutta.

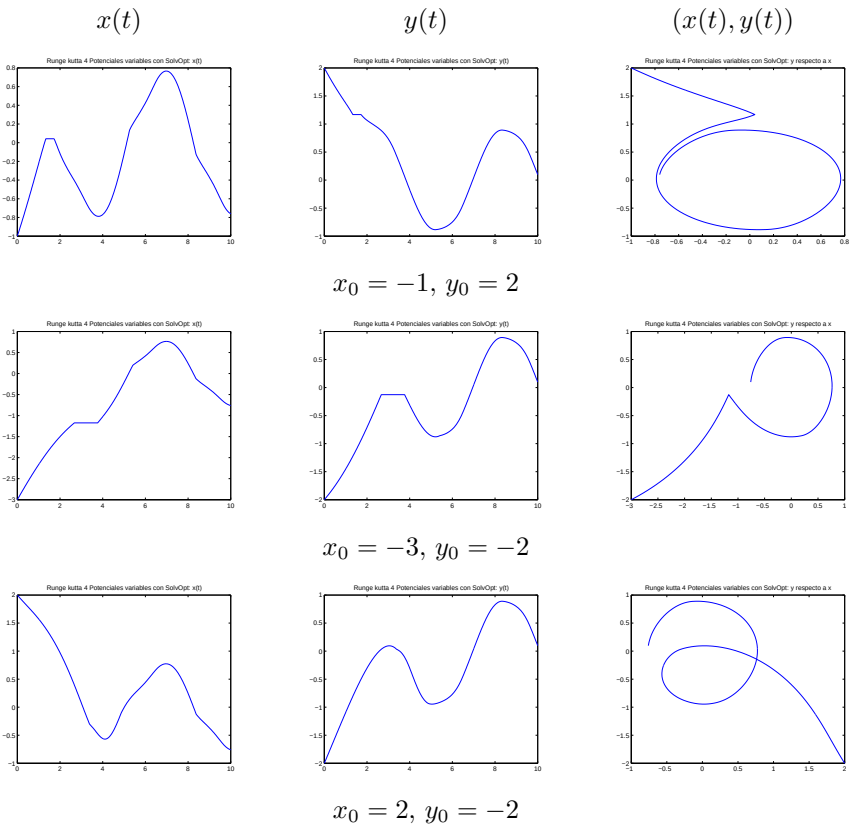



Figura 3.7: Círculos centro y radio variables

**Ejemplo 3.3** Simulamos ahora la solución de la inclusión subdiferencial cuyo potencial es la distancia a la familia de elipses móviles (Figura 3.8)

$$D(t) = \left\{(x,y) \in \mathbb{R}^2 : \frac{(x - \text{sen}(t))^2}{(2 + \cos(t))^2} + (y - (1+t))^2 \leq 1\right\}$$

Los parámetros usados en el código:

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| `a` | `0` | `x0` | `0` |
| `b` | `10` | `y0` | `0` |
| `M` | `1000` | `g1` | `sin(t)` |
| `lambda` | `0.001` | `g2` | `1+t` |

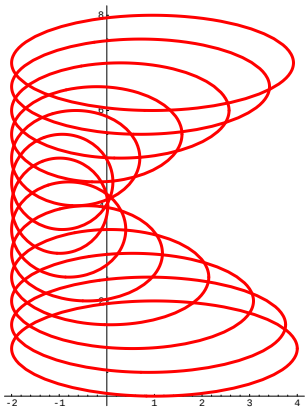
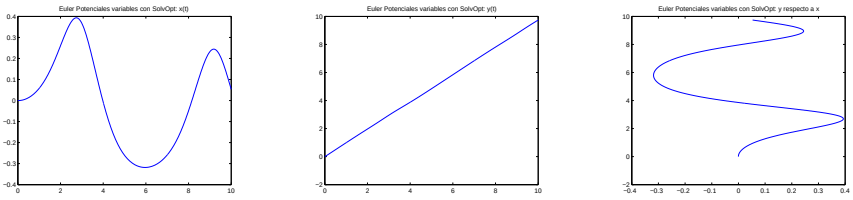


Figura 3.8: Elipses móviles



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

Figura 3.9: Elipses móviles (Euler)

### 3.1.4 Simulaciones con término fuente

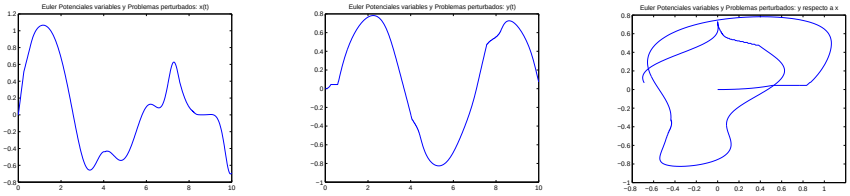
En relación con la estructura de los programas para problemas del tipo (3.1) perturbados con un término fuente (cuyas funciones principales son `eulerSOCD2.m` y `rk4_SOCD2.m`) solamente se tienen las diferencias generales comentadas al inicio de este capítulo sobre el uso del optimizador *SolvOpt*. Debido a que debemos ir evaluando el término fuente, que puede depender del tiempo, para ir calculando los coeficientes $q_i$, éstos y los $k_i$ se deberán calcular a la vez.

**Ejemplo 3.4** Consideremos el problema

$$-\dot{\mathbf{x}}(t) \in \partial d_{C(t)}(\mathbf{x}(t)) - \mathbf{F}(t, \mathbf{x}(t)) \tag{3.4}$$

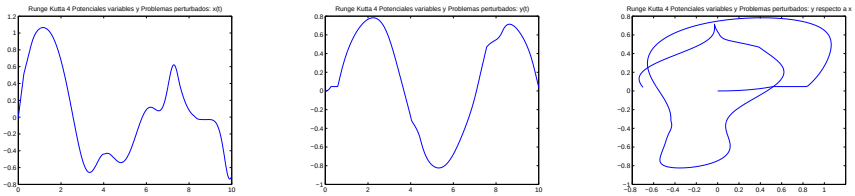
para el caso particular en que $C(t)$ es la familia de conjuntos móviles del Ejemplo 3.1 y $\mathbf{F}(t, x, y) = (\cos(ty), 0)$, con los parámetros

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | x0 | 0 |
| b | 10 | y0 | 0 |
| M | 1000 | g1 | cos(t) |
| lambda | 0.001 | g2 | sin(t) |




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

Figura 3.10: Círculos centro variable y término fuente (Euler)




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

Figura 3.11: Círculos centro variable y término fuente (RK)

Se observan sustanciales diferencias respecto del caso homogéneo. Además, ampliando algunas zonas de las gráficas se aprecian diferencias entre las aproximaciones porporcionadas por los códigos.

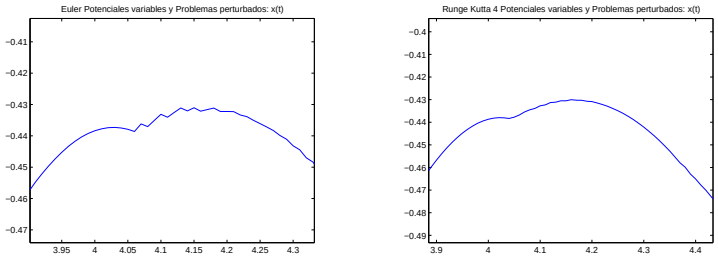



Figura 3.12: Detalle gráfica $x(t)$

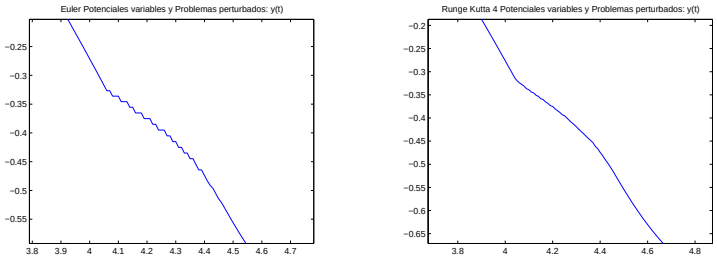


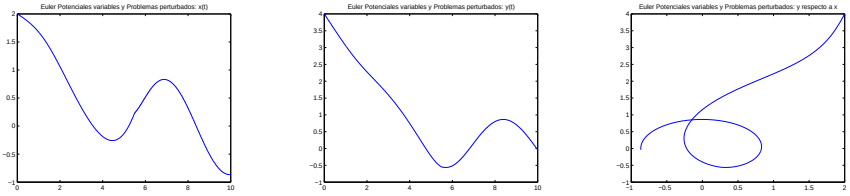

Figura 3.13: Detalle gráfica $y(t)$

Finalmente, cambiamos el término fuente por

$$\mathbf{G}(t,x,y) = \begin{cases} (0,\,0), & \text{si } (x,y) \in C(t) \text{ o } d_{C(t)}(x,y) > 0.1 \\ (-x+\cos(t),\,-y+\text{sen}(t)), & \text{en otro caso} \end{cases}$$

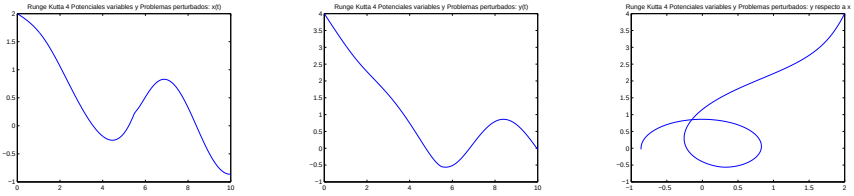
y ejecutamos el código para los parámetros:

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| `a` | `0` | `x0` | `2` |
| `b` | `10` | `y0` | `4` |
| `M` | `1000` | `g1` | `cos(t)` |
| `lambda` | `0.001` | `g2` | `sin(t)` |




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

Figura 3.14: Círculos centro variable y término fuente II (Euler)




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

Figura 3.15: Círculos centro variable y término fuente II (RK)

## 3.2 Problemas bipotenciales

En el caso de problemas bipotenciales del tipo (1.70), resulta interesante considerar potenciales del tipo distancia a un convexo, lo que obliga a modificar los algoritmos y códigos de forma similar a como se ha comentado en el apartado anterior.

Pero, debido a la existencia de dos potenciales, en este caso, deberemos incluir tanto para Euler como Runge-Kutta modificaciones adicionales:

- ✓ Un fichero donde se definirá la función objetivo del segundo potencial.
- ✓ Un fichero donde se realizará la definición de las restricciones del segundo potencial.
- ✓ Dos nuevos parámetros denominados `f1` y `f2` asociados al primer potencial. Los parámetros denominados `g1` y `g2` ya definidos serán los asociados al segundo potencial. Estos parámetros formarán parte de los vectores de inicio de búsqueda del mínimo del optimizador *SolvOpt*.

En cuanto al programa principal del método de Euler, éste poseerá la siguiente modificación con respecto al de problemas bipotenciales presentado en el capítulo anterior:

✓ Se realizará la búsqueda del mínimo para ambos potenciales pero modificando dichas líneas por las correspondientes a el nuevo optimizador *SolvoOpt.*(Ver apartado de potenciales variables).

Para el método de Runge Kutta se han realizado las siguientes modificaciones:

✓ Los dos ficheros asociados al cálculo del mínimo de los dos potenciales se han modificado de forma que haga uso del optimizador *SolvOpt* y devuelva el valor deseado.

✓ Solo en el caso donde los parámetros `f1`,`f2`,`g1` ó `g2` variaran con el tiempo, se debería definir en cada parte del algoritmo donde se calculan los coeficientes $k_i$ y $q_i$ los valores de los parámetros globales utilizados en `puntof` y `puntog`. Debido a que éstos varían con el tiempo, se debe ir repitiendo después de la asignación inicial del valor del parámetro `t` únicamente.

### 3.2.1 Simulaciones

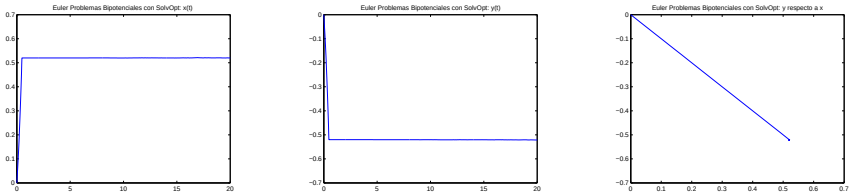
**Ejemplo 3.5** Se considera el problema bipotencial (1.70) con

$$\phi(x,y) = d_{\mathcal{B}}(x,y), \qquad \varphi(x,y) = d_C(x,y), \qquad \beta(t) = (1+t)^2$$

donde $\mathcal{B} = \left\{(x,y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\right\}$ es la bola unidad y $C = [-3/2, -1/2] \times [1/2, 3/2]$ un cuadrado.

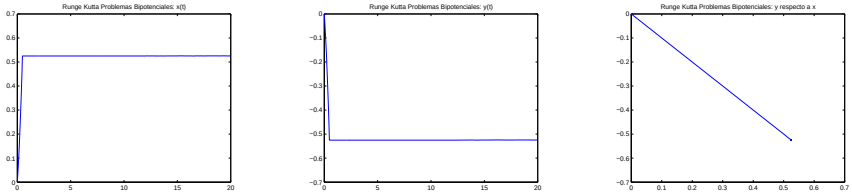
Al ejecutar el código se utilizan los siguiente valores para los parámetros.

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| `a` | `0` | `x0` | `0` |
| `b` | `20` | `y0` | `0` |
| `M` | `1000` | `f1` | `0` |
| `lambda1` | `0.001` | `f2` | `0` |
| | | `lambda2` | `0.001` |




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

Figura 3.16: Bipotencial (Euler)

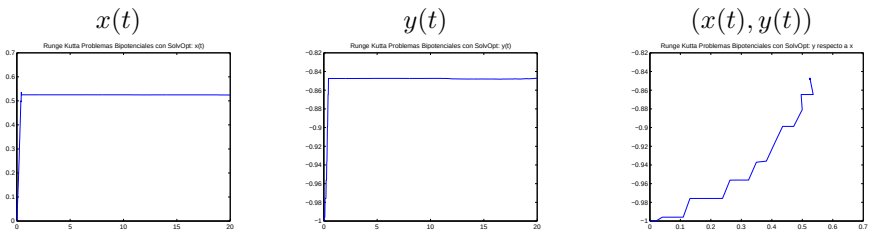



$x(t)$ $y(t)$ $(x(t), y(t))$

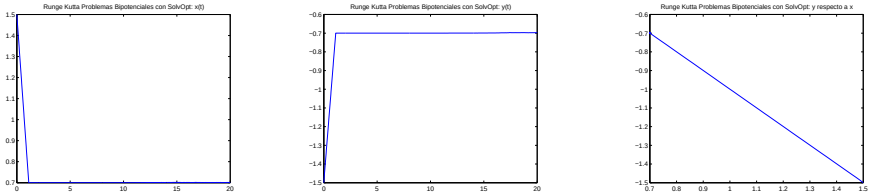
Figura 3.17: Bipotencial (RK)

Del Th. 3.1 en [1] (nuestro ejemplo verifica las hipótesis del mismo) sabemos que cuando $t \to +\infty$, la solución debe convergen a $(0.5, -0.5)$. Nuestra simulación está de acuerdo con esto ya que pata $t = 20$ encontramos el valor $(0.5202, -0.5213)$ con el método de Euler y $(0.5253, -0.5248)$ con Runge-Kutta.

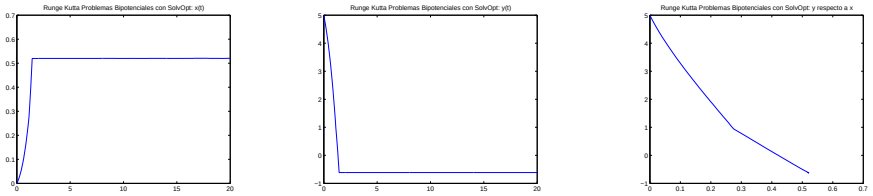
A continuación simulamos el problema cambiando las condiciones iniciales. Presentamos solamente las gráficas obtenidas mediante el código Runge-Kutta, indicando el valor final obtenido y el valor al que converge asintóticamente la solución.




$$x_0 = 0,\ y_0 = -1,\ x(20) = 0.5143 \approx 0.5,\ y(20) = -0.8473 \approx -0.86$$



$$x_0 = 3/2,\ y_0 = -3/2,\ x(20) = 0.7012 \approx 0.707107,\ y(20) = -0.6963 \approx -0.707107$$



$$x_0 = 0,\ y_0 = 5,\ x(20) = 0.5203 \approx 0.5,\ y(20) = -0.6140 \approx -0.5$$

Figura 3.18: Bipotencial (Runge-Kutta)

**Ejemplo 3.6** Nos planteamos ahora un problema de tipo bipotencial en el que uno de los potenciales varía con el tiempo. Cabe decir que esta clase de problemas ha sido menos tratada en la bibliografía que el caso del potencial constante en tiempo. De echo no existen, que nosotros sepamos, resultados teóricos sobre comportamiento asintótico similares a los de [1].

Consideraremos, pues, el problema con

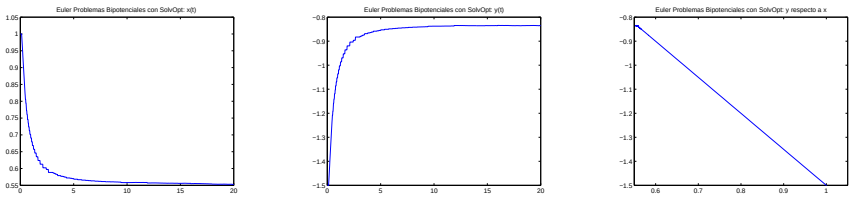
$$\phi(t, x, y) = d_{C(t)}(x, y)$$

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

$$r(t) = 1 + \frac{1}{(1+t)^2}$$

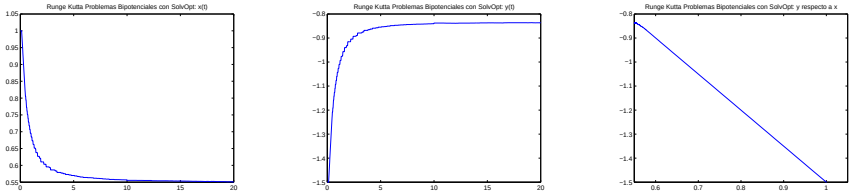
El potencial $\varphi$ y la función $\beta$ son las del ejemplo anterior.

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | x0 | 1 |
| b | 20 | y0 | -3/2 |
| M | 1000 | f1 | 0 |
| lambda1 | 0.001 | f2 | 0 |
| | | lambda2 | 0.001 |




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

Figura 3.19: Bipotencial dependiendo de $t$ (Euler)

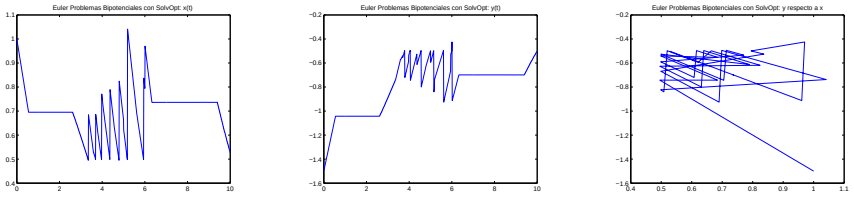



$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 3.20: Bipotencial dependiendo de $t$ (RK)

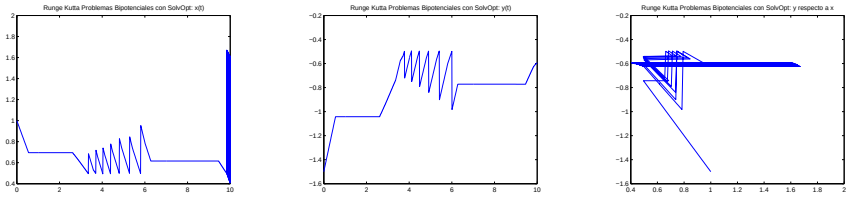
Cambiamos ahora el radio, $r(t) = 1 + \text{sen}(t)/2$ y simulamos para los valores de la tabla.

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | x0 | 1 |
| b | 10 | y0 | -3/2 |
| M | 1000 | f1 | 0 |
| lambda1 | 0.001 | f2 | 0 |
| | | lambda2 | 0.001 |




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

Figura 3.21: Bipotencial dependiendo de $t$ (Euler)




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

Figura 3.22: Bipotencial dependiendo de $t$ (RK)

# Capítulo 4

# Aplicaciones

En este capítulo aplicaremos nuestros códigos a diferentes problemas de interés en ingeniería. En particular se considerarán dos clases de circuitos eléctricos y un sistema físico con fricción de Coulomb (también llamada *dry*).

## 4.1 Análisis de circuitos I

En primer lugar se analizará un circuito RLC no lineal tomado de [15], el cual consta de una resistencia, una bobina, un condensador y un par de diodos zener en serie. Dicho análisis se realizará aplicando conceptos de análisis convexo.

Inicialmente presentaremos su modelo físico, seguidamente desarrollaremos matemáticamente las ecuaciones obtenidas del circuito y finalmente las adaptaremos a nuestros algoritmos para obtener resultados.

### 4.1.1 Modelo físico

Las ecuaciones que describen al circuito (Figura (4.1)) son:

$$E(t) = V_L + V_R + V_Z + V_C,$$
$$I(t) = I_L = I_R = I_Z = I_C = \frac{dq_C}{dt}$$

donde

- $E(t)$ = voltaje aplicado
- $q_C$ = carga del condensador
- $(V_L, I_L)$ = (voltaje,corriente) en la bobina
- $(V_R, I_R)$ = (voltaje,corriente) en la resistencia

- $(V_Z, I_Z)$ = (voltaje,corriente) en la celda que representa a la pareja de diodos zener.
- $(V_C, I_C)$ = (voltaje,corriente) en el condensador

Además debemos tener en cuenta las leyes que regulan el comportamiento de un circuito eléctrico:

$$\begin{aligned} V_L(t) &= L\frac{dI(t)}{dt}, && \texttt{Ley de Lenz} \\ V_R(t) &= RI(t), && \texttt{Ley de Ohm} \\ q_C(t) &= CV_C(t), && \texttt{Carga del condensador} \\ I_Z(t) &= F(V_Z(t)), && \texttt{Corriente en el Zener} \end{aligned}$$

donde `F` es la función característica de un diodo Zener (ver Figura (4.1)).

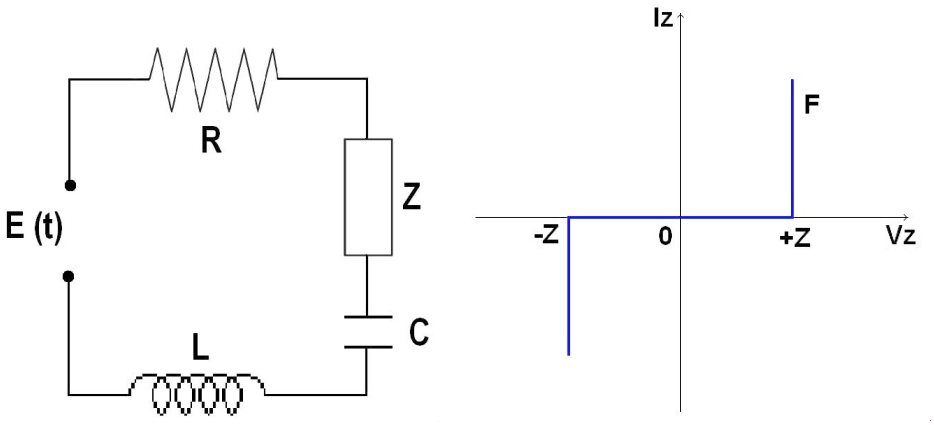



Figura 4.1: Circuito RLC con Zener, Función Característica del Zener

La gráfica `F` es monótona, y se puede ser invertida, obteniendo así la siguiente expresión y gráfica:

$$V_Z = F^{-1}(I_Z) = \begin{cases} +Z & \text{si } I_Z > 0 \\ [-Z, +Z] & \text{si } I_Z = 0 \\ -Z & \text{si } I_Z < 0 \end{cases}$$

En realidad, la obtención de este tipo de respuestas de un elemento no lineal, lleva asociado el uso de un amplificador operacional adecuado que posea una alta ganancia y una impedancia de salida baja, junto con los dos diodos.

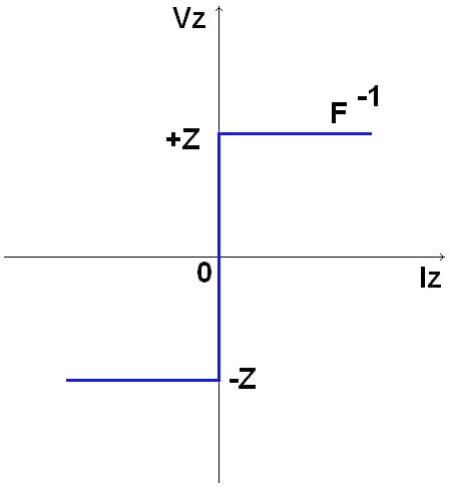



Figura 4.2: Función Característica inversa del Zener

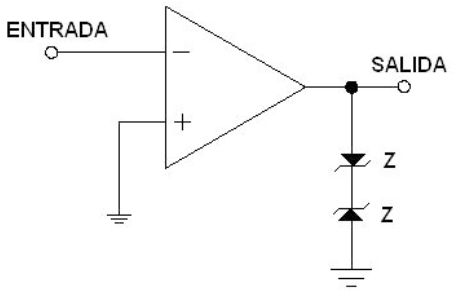



Figura 4.3: Amplificador operacional con diodos Zener

En adición, hay que indicar las condiciones iniciales para completar el estado del circuito. Dichas condiciones son las siguientes:

$$\begin{cases} \dfrac{dq_C(t)}{dt} = I(t), & t > 0 \\ L\dfrac{dI}{dt} + RI + \dfrac{1}{C}q_C(t) + F^{-1}(I(t)) = E(t), & t > 0 \\ q_C(0) = 0, \\ I(0) = 0. \end{cases} \tag{4.1}$$

### 4.1.2 Modelo matemático

Para la interpretación de (4.1) se tiene el siguiente modelo matemático. Considerando una función convexa:

$$f(x) = Z|x|, \mathbf{x} \in \mathbb{R}, \tag{4.2}$$

Entonces tenemos:

$$\partial f(x) = F^{-1}(x), x \in \mathbb{R}, \tag{4.3}$$

y la siguiente formulación de (4.1):

$$[E - L\frac{dI}{dt} - RI - \frac{1}{C}q_C](t) \in \partial f(I(t)), \quad t > 0 \tag{4.4}$$

Teniendo en cuenta el Corolario 1.4 en el Capítulo 1, una alternativa para (4.4) es:

$$[L\frac{dI}{dt} + RI + \frac{1}{C}q_C](t)[j - I(t)] + f(j) - f(I(t)) \geqslant E(t)[j - I(t)], \quad t > 0. \tag{4.5}$$

Esta alternativa es la que habitualmente se utiliza, tanto para análisis teóricos como para experimentos numéricos (ver [15]).

### 4.1.3 Algoritmos Numéricos

Una vez establecido el modelo matemático se realizará una adaptación de nuestros modelos para realizar una simulación numérica. Cabe destacar que nuestros algoritmos permiten resolver de forma directa el problema sin necesidad de utilizar una formulación con desigualdades o técnicas *ad hoc*.

A continuación se presenta la adaptación del problema a nuestros algoritmos realizando diversas aproximaciones.

$$E(t) - L\frac{dI(t)}{dt} - RI(t) - \frac{1}{C}\int_0^t I(s)ds \in Z\partial\phi(I(t)) \tag{4.6}$$

Siguiendo el esquema planteado al inicio del Capitulo 2, sustituimos la subdiferencial por su regularización de Yosida.

$$E(t) - L\frac{dI(t)}{dt} - RI(t) - \frac{1}{C}\int_0^t I(s)ds = \frac{Z}{\lambda}(I(t) - J_\lambda) \tag{4.7}$$

Aproximando la integral aplicando la regla del trapecio y reorganizando se llega a la siguiente ecuación:

$$\frac{dI(t)}{dt} = \frac{-1}{L}(-E(t) + RI(t) + \frac{t}{2C}I(t) + \frac{Z}{\lambda}(I(t) - J_\lambda) = \frac{Z}{\lambda}(I(t) - J_\lambda)) \tag{4.8}$$

Con ello, se puede establecer el siguiente criterio de actualización del valor de la corriente en la iteración actual utilizando Euler del siguiente modo:

$$I^{m,j+1} = I^{m,j} - \frac{Z}{L}\frac{h_m}{\lambda}(I^{m,j} - Y^{m,j}) + h_m\left(\frac{1}{L}E(t^{m,j}) - \frac{R}{L}I^{m,j} - \frac{t^{m,j}}{2LC}I^{m,j}\right) \tag{4.9}$$

donde el término fuente es la expresión siguiente:

$$\frac{1}{L}E(t^{m,j}) - \frac{R}{L}I^{m,j} - \frac{t^{m,j}}{2LC}I^{m,j} \tag{4.10}$$

A la hora de realizar el estudio del circuito RLC, nos interesa conocer su corriente pero también es interesante conocer la carga existente en el condensador. Para encontrar una expresión de la carga adecuada a nuestros algoritmos se parte de:

$$q(t) = \int_0^t I(s)ds \tag{4.11}$$

luego

$$q(t^{m,j+1}) - q(t^{m,j}) = \int_{t^{m,j}}^{t^{m,j+1}} I(s)ds \approx \left(\frac{I(t^{m,j+1}) + I(t^{m,j})}{2}\right)h_m \tag{4.12}$$

volviendo a usar la regla del trapecio para aproximar la integral. Por lo tanto el valor de la carga en el condensador queda como:

$$q(t^{m,j+1}) = q(t^{m,j}) + \frac{h_m}{2}(I(t^{m,j+1}) + I(t^{m,j})) \tag{4.13}$$

### 4.1.4 Simulaciones

En esta subsección se presentan los resultados obtenidos con nuestros algoritmos implementados. Se trata de un caso con término fuente y de una variable, por tanto se hará uso de los códigos programados para el caso de problemas perturbados con una variable.

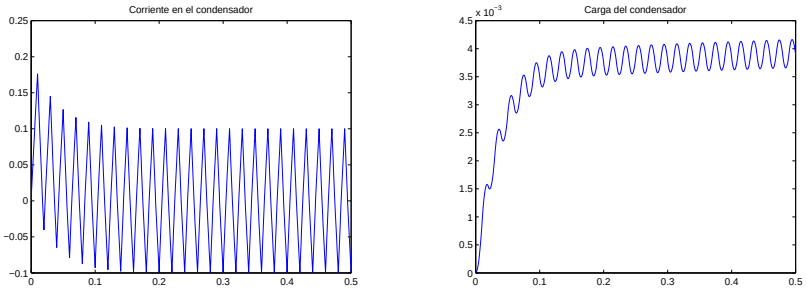
**Ejemplo 4.1** Se considera el potencial $\phi(x) = Z|x|$ y se calcula su solución en el intervalo $[0, 0.5]$ a partir del valor inicial 0 mediante el algorimo de Euler. Los parámetros relativos al circuito que hemos utilizado son los siguientes:

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| L | 0.1 | C | 10 |
| Z | 0.4 | R | 1 |
| H | 0.1 | | |

Se tiene una onda cuadrada de amplitud 2 y frecuencia 50 Hz para la tensión del circuito.

$$E(t) = \begin{cases} 2 & \text{si } t \in [jH, (j+1)H], \quad j \text{ par} \\ 0 & \text{si } t \in [jH, (j+1)H], \quad j \text{ impar} \end{cases}$$

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 0.5 | x0 | 0 |
| M | 1000 | | |




$I(t)$ $q_C(t)$

Figura 4.4: Corriente y carga en el condensador

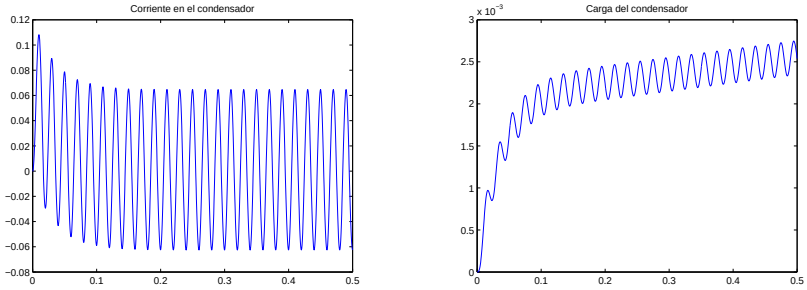
Se observa como la carga del condensador presenta una forma de onda de tipo exponencial creciente característica de la carga de un condensador.

**Ejemplo 4.2** Considerando el mismo potencial que en el ejemplo anterior y calculando la solución en el intervalo $[0, 0.5]$ a partir del valor inicial $0$ mediante el algorimo de Euler.

Los valores de los elementos del circuito no se han modificado, es decir, corresponden con los valores utilizados en el ejemplo anterior. Como tensión en el generador se tiene en este ejemplo una señal senoidal de amplitud 2 y frecuencia 50 Hz.

$$E(t) = 2\sin(2\pi 50t)$$

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 0.5 | x0 | 0 |
| M | 1000 | | |




$I(t)$ $q_C(t)$

Figura 4.5: Corriente y carga en el condensador

## 4.2 Análisis de circuitos II

### 4.2.1 Modelo físico

En [14] se plantea el estudio de un circuito RCL de carácter no lineal, donde la no linealidad está asociado a un fenómeno de rotura del dieléctrico.

El circuito bajo estudio (Figura (4.6)) consta de una resistencia `R`, una inductancia `L` y un condensador `C` en paralelo con un arco eléctrico de voltaje crítico igual a $V_a$.

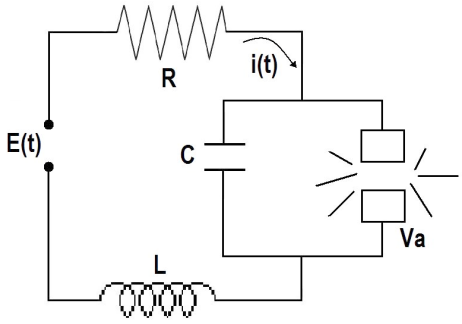



Figura 4.6: Circuito RLC con arco eléctrico

La notación se resume en las siguientes tablas:

| Constantes | |
|---|---|
| $R$ | Resistencia |
| $L$ | Inductancia |
| $C$ | Capacitancia |
| $V_a$ | Voltaje crítico del arco |

| Funciones | |
|---|---|
| $E(t)$ | Voltaje aplicado |
| $q(t)$ | Carga total |
| $i(t)$ | Intensidad |
| $q_C(t)$ | Carga del condensador |
| $V_C(t)$ | Voltaje en el condensador |
| $j(V_C)$ | Corriente del arco |

El subcircuito formado por el condensador y el arco eléctrico simula el dieléctrico del condensador cerca de la rotura del dieléctrico. Para ver si se está en anterior situación, hay que comprobar que el voltaje del condensador satisfaga la siguiente condición:

$$-V_a \leq V_c(t) \leq V_a, \quad t \geq 0, \tag{4.14}$$

El arco eléctrico presente en el circuito posee la función característica $j = j(V_C)$ mostrada en la figura (4.2.1). Esta función se puede lograr con un par de diodos Zener, por ejemplo y si queremos realizar una descripción mas detallada necesitamos saber más aparte de los conceptos básicos de la función.

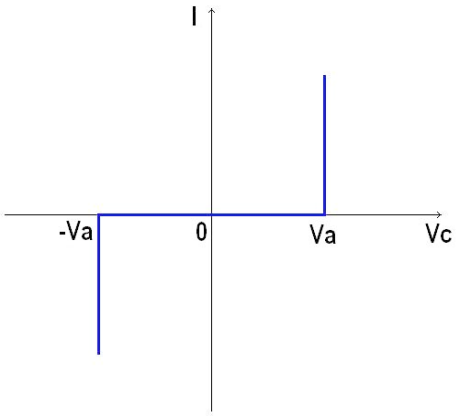



Figura 4.7: Función Característica del arco eléctrico

En términos analíticos, sabiendo que $V_C$ es función del tiempo, tenemos:

$$j(V_C) = \begin{cases} 0 & \text{si } |V_C| < V_a \\ 0 & \text{si } V_C = V_a \text{ y } \frac{dV_C}{dt} < 0 \\ 0 & \text{si } V_C = -V_a \text{ y } \frac{dV_C}{dt} > 0 \\ \mathbb{R}^+ & \text{si } V_C = V_a \text{ y } \frac{dV_C}{dt} = 0 \\ \mathbb{R}^- & \text{si } V_C = -V_a \text{ y } \frac{dV_C}{dt} = 0 \end{cases}$$

Dentro de los límites definidos en (4.14), la función característica del condensador posee la relación lineal que tiene habitualmente:

$$q_C(t) = CV_C(t)$$

Finalmente las relaciones existentes en el circuito vendrán dadas por las Leyes de Kirchoff:

$$\begin{cases} E(t) - L\dfrac{di}{dt}(t) - Ri(t) - V_C(t) = 0, t > 0 \\ i(t) - C\dfrac{dV_C}{dt}(t) - j(V_C(t)) = 0, t > 0 \end{cases} \tag{4.15}$$

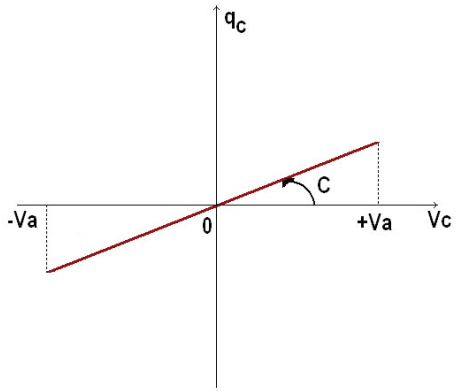



Figura 4.8: Carga del condensandor frente a la tensión

La aplicación que estamos estudiando se basa en la compatibilización de las ecuaciones (4.14)-(4.15), mas unas condiciones iniciales.

En el siguiente apartado se presenta de forma breve el modelo matemático planteado y seguidamente la adaptación de nuestros algoritmos a este problema.

### 4.2.2 Modelo matemático

Se denota por $I_a = [-V_a, V_a] \subset \mathbb{R}$ y se tiene que $j(V_C) = \partial J_{I_a}(V_C)$, donde

$$J_{I_a}(x) = \begin{cases} 0, & x \in I_a \\ +\infty, & x \notin I_a \end{cases}$$

es la función indicatriz del intervalo $I_a$ (que nosotros hemos denotado en el Capítulo 1 por $\psi_{I_a}$).

Aplicando las leyes de Kirchoff (4.15) a las dos mallas del circuito, se llega al sistema de ecuaciones:

$$\begin{cases} E(t) - L\dfrac{di}{dt}(t) - Ri(t) - V_C(t) = 0 \\ i(t) - C\dfrac{dV_C}{dt}(t) \in \partial J_{I_a}(V_C(t)) \end{cases} \tag{4.16}$$

para $t > 0$, con las condiciones iniciales $i(0) = V_C(0) = 0$.

### 4.2.3 Algoritmos Numéricos

Definiendo las nuevas variables $x(t) = i(t)$, $y(t) = V_C(t)$, el potencial convexo

$$\phi(x, y) = J_{I_a}(y)$$

y el término fuente

$$F(t, x, y) = -\left(\frac{1}{L}\left(Rx + y - E(t)\right),\ \frac{-x}{C}\right)$$

las ecuaciones (4.16) del circuito pueden reescribirse como una inclusión subdiferencial con término fuente en la forma estándar que hemos manejado en la memoria,

$$-\left(\dot{x}(t), \dot{y}(t)\right) \in \partial\phi(x(t), y(t)) - F(t, x(t), y(t))$$

A la hora de resolver el problema numéricamente, serviría el código de dos variables, teniendo en cuenta que la resolvente de Moreau sería de la forma

$$\begin{aligned} J_\lambda(x, y) &= \operatorname*{argmin}_{(u,v)\in\mathbb{R}^2} J_{I_a}(v) + \frac{1}{2\lambda}\left((u - x)^2 + (v - y)^2\right) \\ &= \operatorname*{argmin}_{u\in\mathbb{R}, v\in I_a} \frac{1}{2\lambda}\left((u - x)^2 + (v - y)^2\right) = (x, \mathrm{proj}(y; I_a)) \end{aligned}$$

La proyección ortogonal sobre un intervalo es fácil de calcular, por lo que tenemos la siguiente expresión expícita para la resolvente

$$J_\lambda(x, y) = \begin{cases} (x, y), & -V_a \le y \le V_a \\ (x, V_a), & y > V_a \\ (x, -V_a), & y < -V_a \end{cases}$$

Dicha resolvente estará definida de forma que no será necesario el uso de ningún algoritmo de búsqueda de mínimos.

En relación a la aproximación para el cálculo de la carga en el circuito, deberemos tener en cuenta (4.13).

### 4.2.4 Simulaciones

Una vez establecido el tipo de problema matemáticamente y adecuado a nuestros algoritmos podremos representar algún caso práctico. Teniendo en cuenta el tipo de problema, haremos uso de los algoritmos asociados a problemas perturbados para dos variables con Runge Kutta 4. En cuanto a resultados, nuestro interés es la de obtener el valor de la corriente en el circuito, el voltaje en el condensador y la carga en el circuito. A continuación se presentan varios casos de nuestro problema.

**Ejemplo 4.3** Los parámetros relativos al circuito que hemos utilizado son los siguientes:

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| L | 1 | C | 1 |
| R | 1 | Va | 2.5 |

Como tensión se tiene una exponencial creciente, cuya expresión es la siguiente:

$$E(t) = 5e^t$$

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 3 | x0 | 0 |
| M | 1000 | y0 | 0 |

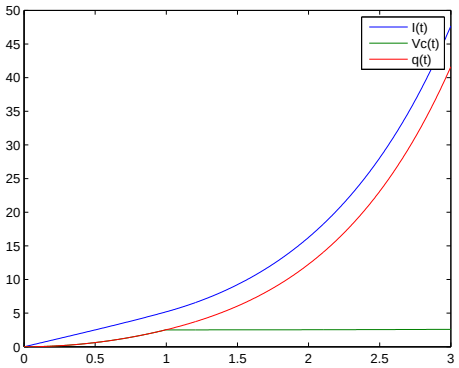



Figura 4.9: Corriente y carga en el condensador

**Ejemplo 4.4** Los parámetros relativos a este ejemplo, tenemos:

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| L | 1 | C | 1 |
| R | 1 | Va | 1 |

En este caso se excita el circuito con una señal senoidal de amplitud 20 y pulsación 10.

$$E(t) = 20\sin(10t)$$

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 3 | x0 | 0 |
| M | 1000 | y0 | 0 |

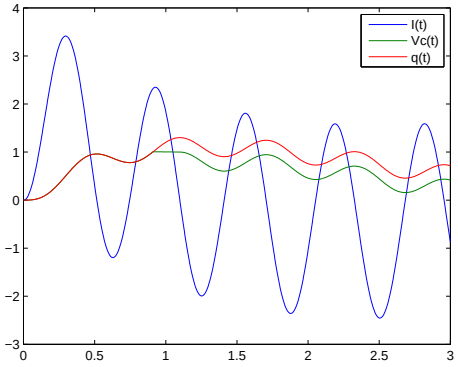



Figura 4.10: Corriente y carga en el condensador

**Ejemplo 4.5** Se excita de nuevo con una señal tipo seno pero aumentando su amplitud hasta un valor de 100.

$$E(t) = 100\sin(10t)$$

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| L | 1 | C | 1 |
| R | 1 | Va | 1 |

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 3 | x0 | 0 |
| M | 1000 | y0 | 0 |

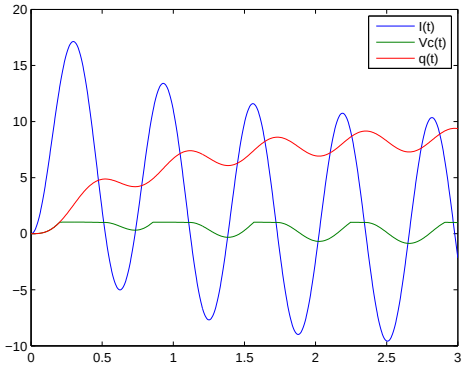



Figura 4.11: Corriente y carga en el condensador

**Ejemplo 4.6** Como tensión del circuito se aplica una onda cuadrada de la siguiente forma:

$$E(t) = \begin{cases} 100 & \text{si } t \in [j, j+1],\ j \text{ par} \\ -100 & \text{si } t \in [j, j+1],\ j \text{ impar} \end{cases}$$

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| L | 1 | C | 1 |
| R | 1 | Va | 1 |

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 6 | x0 | 0 |
| M | 1000 | y0 | 0 |

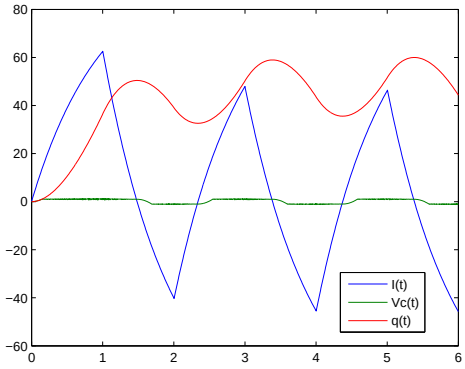



Figura 4.12: Corriente y carga en el condensador

**Ejemplo 4.7** Como tensión del circuito se aplicará una onda cuadrada con las mismas características que en el ejemplo anterior hasta t = 3 segundos, donde a partir de ahí no se excitará el circuito con ninguna señal.

```
if (t<3)
    Et =(100*square(2*pi*0.5*t));
else
    Et = 0;
end
```

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| L | 1 | C | 1 |
| R | 1 | Va | 1 |

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 6 | x0 | 0 |
| M | 1000 | y0 | 0 |

En el instante t = 3 segundos se observa que la corriente comienza a disminuir de forma exponencial y en cambio la carga aumenta hasta llegar a un valor fijo.

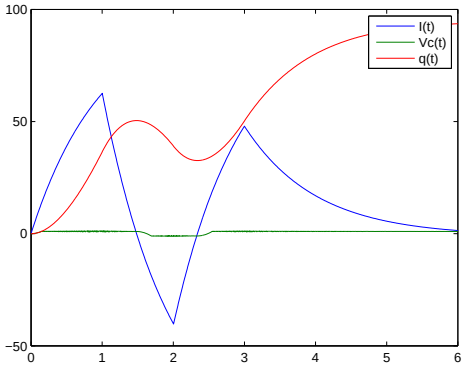



Figura 4.13: Corriente y carga en el condensador

## 4.3 Sistema mecánico con fricción de Coulomb

### 4.3.1 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 4.14 se observa un dispositivo de esa naturaleza.

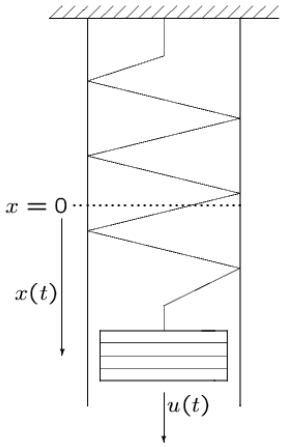



Figura 4.14: Dispositivo con fricción de Coulomb

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

$$m\ddot{x}(t) = F \tag{4.17}$$

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

$$-kx(t) \tag{4.18}$$

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

$$-r\dot{x}(t) \tag{4.19}$$

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

$$-c\,\mathrm{Sgn}(\dot{x}(t)) \tag{4.20}$$

donde

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

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

$$F = p(t) - kx(t) - c\dot{x}(t) - c\,\mathrm{Sgn}(\dot{x}(t)) \tag{4.21}$$

que junto con (4.17) nos la inclusión de segundo orden

$$m\ddot{x}(t) + kx(t) + r\dot{x}(t) - p(t) \in -c\,\mathrm{Sgn}(x(t)) \tag{4.22}$$

Evidentemente, si $\phi(z) = |z|$, sabemos que $\mathrm{Sgn} = \partial\phi$, luego tenemos la forma subdiferencial de la ecuación que rige el estado de un sistema de este tipo

$$-m\ddot{x}(t) - kx(t) - r\dot{x}(t) + p(t) \in c\partial\phi(\dot{x}(t)) \tag{4.23}$$

### 4.3.2 Modelo matemático

Obviamente (4.23) puede escribirse como un sistema de la forma

$$\begin{cases} \dot{x}(t) = y(t) \\ \dot{y}(t) \in \frac{1}{m}\left(-kx(t) - r\dot{x}(t) + p(t)\right) - \frac{c}{m}\partial\phi(y(t)) \end{cases} \tag{4.24}$$

y, si definimos el término fuente

$$\mathbf{F}(t, x, y) = \left(y,\ \frac{-k}{m}x + \frac{-r}{m}y + \frac{1}{m}p(t)\right)$$

y el potencial convexo $\Phi(x, y) = |y|$, tenemos escrita la ecuación del sistema en la forma estándar de una inclusión diferencial de dos variables con término fuente:

$$-(\dot{x}(t), \dot{y}(t)) \in \partial\Phi(x(t), y(t)) - \mathbf{F}(t, x(t), y(t)) \tag{4.25}$$

### 4.3.3 Algoritmos Numéricos

Podemos, pues, usar los algoritmos que hemos expuesto en el segundo capítulo 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 $(x, y) \in \mathbb{R}^2$:

$$\operatorname*{argmin}_{(z_1,z_2)\in\mathbb{R}^2} \Phi(z_1, z_2) + \frac{1}{2\lambda}\left((z_1 - x)^2 + (z_2 - y)^2\right) =$$

$$\operatorname*{argmin}_{(z_1,z_2)\in\mathbb{R}^2} |z_2| + \frac{1}{2\lambda}\left((z_1 - x)^2 + (z_2 - y)^2\right) =$$

$$\left(x, \operatorname*{argmin}_{z_2\in\mathbb{R}} |z_2| + \frac{1}{2\lambda}(z_2 - y)^2\right)$$

### 4.3.4 Simulaciones

Se presentan seguidamente diversas simulaciones.

**Ejemplo 4.8** Los parámetros relativos al problema de fricción de Coulomb (tomados de [7]) son los siguientes:

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| KHooke | 1 | masa | 1 |
| Kroz | 0.2 | Kdry | 4 |

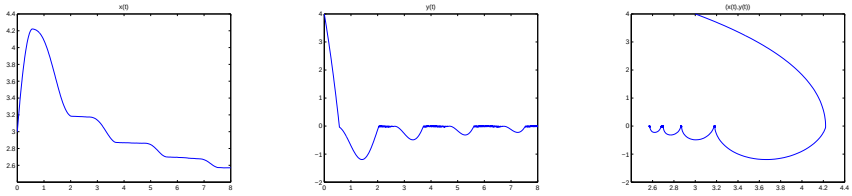
donde dichos parámetros poseen la siguiente correspondencia con los utilizados en el apartado de simulación numérica:

- KHooke $= k$
- Kroz $= \beta$
- masa $= m$
- Kdry $= \gamma$

La fuerza exterior que actúa sobre el muelle tiene la siguiente expresión:

$$p(t) = 2\sin(\pi t)$$

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 8 | x0 | 3 |
| M | 1000 | y0 | 4 |



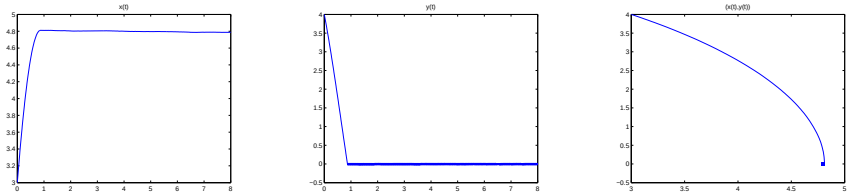
$x(t)$ $y(t)$ $(x(t), y(t))$

Figura 4.15: Posición, velocidad y trayectoria del muelle

**Ejemplo 4.9** Los parámetros relativos al problema de dry friction son los mismos que en el ejemplo anterior con la salvedad de que en este caso la fuerza exterior tendrá un valor constante.

$$p(t) = 4;$$

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 8 | x0 | 3 |
| M | 1000 | y0 | 4 |



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

Figura 4.16: Posición, velocidad y trayectoria del muelle para fuerza constante

**Ejemplo 4.10** Los parámetros relativos al problema de dry friction son los siguientes:

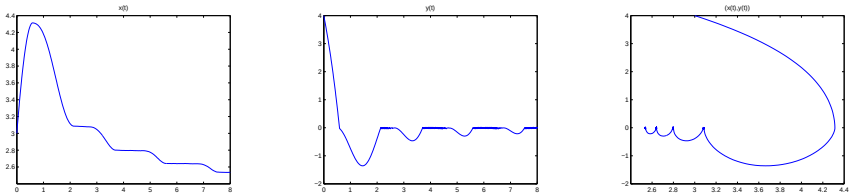
| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| KHooke | 1 | masa | 1 |
| Kroz | 0 | Kdry | 4 |

En este ejemplo no existe rozamiento de la masa del muelle en el medio donde se encuentre.

La fuerza exterior que actúa sobre el muelle tiene la siguiente expresión:

$$p(t) = 2\sin(\pi t)$$

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 8 | x0 | 3 |
| M | 1000 | y0 | 4 |



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

Figura 4.17: Posición, velocidad y trayectoria del muelle sin rozamiento

Obtenemos curvas muy similares al primer ejemplo estudiado en esta aplicación. Esto es debido a que el valor de Kroz utilizado en ambos ejemplos poseen poca diferencia entre sí.

**Ejemplo 4.11** Los parámetros relativos al problema de dry friction son los siguientes:

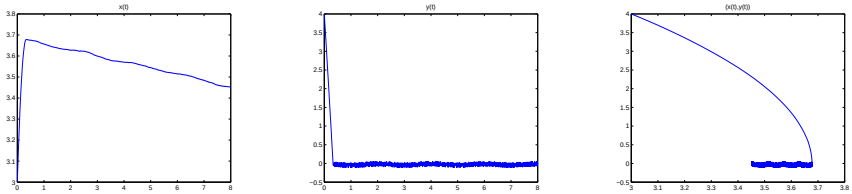
| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| KHooke | 1 | masa | 1 |
| Kroz | 0.2 | Kdry | 10 |

Se ha aumentado el parámetro asociado al rozamiento seco.
La fuerza exterior que actúa sobre el muelle tiene la siguiente expresión:

$$p(t) = 2\sin(\pi t)$$

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 8 | x0 | 3 |
| M | 1000 | y0 | 4 |

En la gráfica relativa a la posición del muelle, se puede apreciar que éste apenas se mueve a lo largo del tiempo. Además la velocidad disminuye de forma favorable haciéndose casi nula en un corto periodo de tiempo. Esto es debido a que el rozamiento debido al fenómeno de dry friction es alto.



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

Figura 4.18: Posición, velocidad y trayectoria del muelle con un rozamientro dry mayor

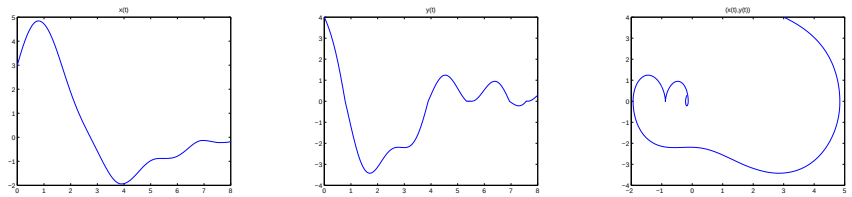
**Ejemplo 4.12** Los parámetros relativos al problema de dry friction son los siguientes:

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| KHooke | 1 | masa | 1 |
| Kroz | 0.2 | Kdry | 1 |

En este ejemplo se ha disminuido el valor de `Kdry`.
La fuerza exterior que actúa sobre el muelle tiene la siguiente expresión:

$$p(t) = 2\sin(\pi t)$$

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| a | 0 | lambda | 0.001 |
| b | 8 | x0 | 3 |
| M | 1000 | y0 | 4 |



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

Figura 4.19: Posición, velocidad y trayectoria del muellecon un rozamientro dry menor

Con un valor tan pequeño de `Kdry`, se tienen más oscilaciones en la trayectoria del muelle.

# Capítulo 5

# Conclusiones y Expectativas

En el presente trabajo se han propuesto e implementado en MATLAB algoritmos para resolver sistemas de ecuaciones diferenciales asociados a potenciales convexos no necesariamente diferenciables mediante la combinación de la regularización de Yosida con un método numérico para ecuaciones diferenciales ordinarias. Dichos algoritmos son válidos incluso en el caso de potenciales variables a lo largo del tiempo o cuando existen términos fuente perturbando la subdiferencial. También para problemas más complicados, como los llamados bipotenciales.

Cabe comentar que la labor de programación ha sido especialmente laboriosa, ya que se han realizado múltiples modificaciones a partir del código original de forma gradual. Cada uno de los códigos presentados en la memoria ha sido ampliamente testado con diferentes problemas.

Aparte de los ejemplos usados más o menos académicos hemos usado con éxito nuestra aproximación con diversas aplicaciones prácticas, en particular con dos modelos de circuitos RLC no lineales y un sistema mecánico con fricción de Coulomb.

En cuanto a expectativas y futuros desarrollos del tema cabe reseñar los siguientes puntos:

- Aunque en nuestros algoritmos solamente se haya hecho uso de dos métodos numéricos (Euler y Runge-Kutta), la metodología que planteamos admite la utilización de cualquier método numérico para ecuaciones diferenciales ordinarias combinado con la regularización de Yosida, por lo que sería interesante proseguir las investigaciones en este aspecto.

- Asimismo, aunque nos hemos restringido al caso bidimensional, los esquemas han sido planteados para un número arbitrario de dimensiones, con lo que las únicas dificultades para abordar problemas de mas de dos dimensiones son de carácter computacional.

- Los códigos programados se pueden utilizar de forma fiable ya que han sido ampliamente testados, lo que abre expectativas para adaptarlos al

tratamiento de problemas complicados que usualmente se formulan de forma simplificada. Así pensamos que podría ser útil, por ejemplo, para simular de forma sistemática el comportamiento de circuitos que contengan diodos, circuitos con diodos zener, en definitiva circuitos que posean algún elemento no lineal.

- Aunque la metodología que proponemos solamente es válida para potenciales convexos, en otros casos (por ejemplo potenciales localmente Lipschitz) permitiría resolver el problema convexificado o relajado, que puede aportar información sobre el original.

# Capítulo 6

# Anexos

## 6.1 Algoritmos originales

### 6.1.1 EULER para una variable

```
function E = euler(f,a,b,ya,M) \vspace{-0.1cm} % espacio entre lineas
% Datos
% - f es la funcion, almacenada como una cadena de caracteres 'f'
% - a y b son los extremos derecho e izquierdo del intervalo
% - ya es la condicion inicial y(a)
% - M es el numero de pasos
% Resultado
% - E = [T' X'] siendo T el vector de las abscisas e X el vector de las ordenadas
 h =(b-a)/M;
 T = zeros(1,M+1);
 X = zeros(1,M+1);
 T = a:h:b;
 X(1)=ya;
 for j=1:M
   X(j+1) = X(j) + h*feval(f,T(j),X(j));
 end
 E = [T' X'];
```

### 6.1.2 EULER para dos variables

```
 function [E,F] = eulersist(f,g,a,b,ya,ya2,M) \\
% Datos
% - f es la funcion, almacenada como una cadena de caracteres 'f'
% - g es la otra funcion del sistema, almacenada como una cadena de caracteres 'g'
% - a y b son los extremos derecho e izquierdo del intervalo
% - ya es la condicion inicial y(a) de la funcion f
```

```
% - ya2 es la condicion inicial y(a) de la funcion g
% - M es el numero de pasos
% Resultado
% - Solucion de f: E = [T' X'];
% siendo T el vector de las abscisas e X el vector de las ordenadas
% - Solucion de g: F = [T' Y'];
% siendo T el vector de las abscisas e Y el vector de las ordenadas
 h =(b-a)/M;
 T = zeros(1,M+1);
 X = zeros(1,M+1);
 Y = zeros(1,M+1);
 T = a:h:b;
 X(1)=ya;
 Y(1)= ya2;
 for j=1:M
 \hspace*{0.2cm} X(j+1) = X(j) + h*feval(f,T(j),X(j),Y(j));
 \hspace*{0.2cm} Y(j+1) = Y(j) + h*feval(g,T(j),X(j),Y(j));
 end
 E = [T' X'];
 F = [T' Y'];
```

### 6.1.3 RUNGE KUTTA 4 para una variable

```
function R= rk4(f,a,b,ya,M)
% Datos
% - f es la funcion, almacenada como una cadena de caracteres 'f'
% - a y b son los extremos derecho e izquierdo del intervalo
% - ya es la condicion inicial y(a)
% - M es el numero de pasos
% Resultado
% - R = [T' Y'] siendo T el vector de las abscisas e Y el vector de las ordenadas
 h =(b-a)/M;
 T = zeros(1,M+1);
 Y = zeros(1,M+1);
 T = a:h:b;
 Y(1)=ya;
 f = T-Y;
 for j=1:M
     k1 = h.*feval('f',T(j),Y(j));
     k2 = h.*feval('f',T(j)+h/2,Y(j)+k1/2);
     k3 = h.*feval('f',T(j)+h/2,Y(j)+k2/2);
     k4 = h.*feval('f',T(j)+h,Y(j)+k3);
     Y(j+1)= Y(j) + (k1+2*k2+2*k3+k4)/6;
 end
```

```
 R=[T' Y'];
```

### 6.1.4 RUNGE KUTTA 4 para dos variables

```
% Runge - kutta 4 para sistemas:

function [R,S]= rk4sistemas(f,g,a,b,ya,ya2,M)

% Datos
% - f es una funcion, almacenada como una cadena de caracteres 'f'
% - g es la otra funcion del sistema, almacenada como una cadena de caracteres 'g'
% - a y b son los extremos derecho e izquierdo del intervalo
% - ya es la condicion inicial y(a) de la funcion f
% - ya2 es la condicion inicial y(a) de la funcion g
% - M es el numero de pasos
%  Resultado
% - Solucion de f: R = [T' X'];
% siendo T el vector de las abscisas e Y el vector de las ordenadas
% - Solucion de g: S = [T' Y'];
% siendo X el vector de las abscisas y Z el vector de las ordenadas

h = (b-a)/M;

T = zeros(1,M+1);
% funcion f
X = zeros(1,M+1);
% funcion g
Y = zeros(1,M+1);

T=a:h:b;
X(1)=ya;
Y(1)=ya2;

for j=1:M
        f1 = feval(f,T(j),X(j),Y(j));
        g1 = feval(g,T(j),X(j),Y(j));
        f2 = feval(f,T(j)+(h/2),X(j)+(h/2)*f1,Y(j)+(h/2)*g1);
        g2 = feval(g,T(j)+(h/2),X(j)+(h/2)*f1,Y(j)+(h/2)*g1);
        f3 = feval(f,T(j)+(h/2),X(j)+(h/2)*f2,Y(j)+(h/2)*g2);
        g3 = feval(g,T(j)+(h/2),X(j)+(h/2)*f2,Y(j)+(h/2)*g2);
        f4 = feval(f,T(j)+(h/2),X(j)+(h/2)*f3,Y(j)+(h/2)*g3);
        g4 = feval(g,T(j)+(h/2),X(j)+(h/2)*f3,Y(j)+(h/2)*g3);
        X(j+1)= X(j) + (h/6)*(f1+2*f2+2*f3+f4);
        Y(j+1)= Y(j) + (h/6)*(g1+2*g2+2*g3+g4);
end
```

```
% resultados
R=[T' X'];
S=[T' Y'];
```

## 6.2 Algoritmos básicos

### 6.2.1 EULER para una variable

```
function E1 = eulerB(a,b,x0,M)
% Metodo de Euler (algoritmo bico) para una variable

global cx lambda
% Datos
%         - a y b son los extremos derecho e izquierdo del intervalo
%         - x0 es la condicion inicial y(a)
%         - M es el numero de pasos
%  Resultado
%         - E1 = [T' X'] siendo T el vector de las abscisas e X el vector
%           de las ordenadas

h =(b-a)/M;
lambda = 0.001;
T = zeros(1,M+1);
X = zeros(1,M+1);
Min = zeros(1,M+1);
T = a:h:b;
X(1)=x0;

for j=1:M
    if j==1
        cx=x0;
    else
        cx = X(j);
    end
    [minimo,fval] = fminsearch(@phi2,cx);
    Min(j) = minimo;
    X(j+1) = X(j) - (h/lambda)*(X(j)-Min(j));
end

E1 = [T' X'];
```

### 6.2.2 EULER para dos variables

```
function [E1,E2] = eulerB2(a,b,x0,y0,M)
% Metodo de Euler (algoritmo bico) para dos variables.

global cx cy lambda
% Datos
%         - a y b son los extremos derecho e izquierdo del intervalo
%         - x0 es la condicion inicial para la variable x
%         - y0 es la condicion inicial para la variable y
%         - M es el numero de pasos
%  Resultado
%         - E1 = [T' X'] siendo T el vector de las abscisas e X el vector
%           de las ordenadas
%         - E2 = [T' Y'] siendo T el vector de las abscisas e Y el vector
%         de las ordenadas.

h =(b-a)/M;
lambda = 0.001;
T = zeros(1,M+1);
X = zeros(1,M+1);
Y = zeros(1,M+1);
T = a:h:b;
X(1) = x0;
Y(1) = y0;

for j=1:M
    if j==1
        cx = x0;
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
    end
    % Calculo del minimo de la funcion phi2
    [minimo,fval] = fminsearch(@phi2,[cx,cy]);
    % Actualizaci de X y Y : obtencion de las soluciones
    X(j+1) = X(j) - (h/lambda)*(X(j)-minimo(1));
    Y(j+1) = Y(j) - (h/lambda)*(Y(j)-minimo(2));
end

% Creacion de las matrices solucion E1 y E2
E1 = [T' X'];
E2 = [T' Y'];
```

### 6.2.3 RUNGE KUTTA 4 para una variable

```
% Metodo de Runge kutta 4 (algoritmo bico) para una variable.
function [RK1]= rk4_B(a,b,x0,M)

global lambda minimox puntox

h = (b-a)/M;
lambda = 0.001;

T = zeros(1,M+1);
T = a:h:b;
X = zeros(1,M+1);
X(1) = x0;

for j=1:M
    if j==1
        cx = x0;
    else
        cx = X(j);
    end

    puntox = cx;
    minimox = calculamin(puntox);
    k1 = feval('yo',puntox);
    puntox = cx + ((h*k1)/2);
    minimox = calculamin(puntox);
    k2 = feval('yo',puntox);
    puntox = cx + ((h*k2)/2);
    minimox = calculamin(puntox);
    k3 = feval('yo',puntox);
    puntox = cx + h*k3;
    minimox = calculamin(puntox);
    k4 = feval('yo',puntox);
    X(j+1)= X(j) + (h/6)*(k1+2*k2+2*k3+k4);
end

RK1=[T' X'];
```

### 6.2.4 Función Yoshida para una variable

```
function p = yo(x)
% Funcion Yoshida para una variable.
global lambda minimox
p = -(1/lambda)*(x(1)-minimox);
```

### 6.2.5 Función calculamin para una variable

```
function minimo = calculamin(k)
[min,fval] = fminsearch(@phi2,k);
minimo = min;
```

### 6.2.6 RUNGE KUTTA 4 para dos variables

```
% Metodo de Runge kutta 4 (algoritmo bico) para dos variables.
function [RK1,RK2]= rk4_B2(a,b,x0,y0,M)

global cx cy lambda minimo puntox puntoy

h = (b-a)/M;
lambda = 0.001;

T = zeros(1,M+1);
T = a:h:b;
X = zeros(1,M+1);
Y = zeros(1,M+1);

X(1) = x0;
Y(1) = y0;

for j=1:M
    if j==1
        cx = x0;
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
    end

    puntox = cx;
    puntoy= cy;
    punto = [puntox,puntoy];
    minimo = calculamin2(punto);
    k1 = feval('yo2',punto);
    puntox = cx + ((h*k1(1))/2);
    puntoy = cy + ((h*k1(2))/2);
    punto = [puntox,puntoy];
    minimo = calculamin2(punto);
    k2 = feval('yo2',punto);
    puntox = cx + ((h*k2(1))/2);
    puntoy = cy + ((h*k2(2))/2);
    punto = [puntox,puntoy];
```

```
    minimo = calculamin2(punto);
    k3 = feval('yo2',punto);
    puntox = cx + h*k3(1);
    puntoy = cy + h*k3(2);
    punto = [puntox,puntoy];
    minimo = calculamin2(punto);
    k4 = feval('yo2',punto);
    X(j+1)= X(j) + (h/6)*(k1(1)+2*k2(1)+2*k3(1)+k4(1));
    Y(j+1)= Y(j) + (h/6)*(k1(2)+2*k2(2)+2*k3(2)+k4(2));
end

RK1 = [T' X'];
RK2 = [T' Y'];
```

### 6.2.7 Función Yoshida para dos variables

```
function p = yo2(x)
% Funcion Yoshida para dos variables.
global lambda minimo
yo_1 = -(1/lambda)*(x(1)-minimo(1));
yo_2= -(1/lambda)*(x(2)-minimo(2));
p = [yo_1,yo_2];
```

### 6.2.8 Función calculamin2 para dos variables

```
function [mini] = calculamin2(k)
[min,fval] = fminsearch(@phi2,k);
mini = min;
```

## 6.3 Algoritmos para potenciales variables

### 6.3.1 EULER para una variable

```
function E1 = eulerC(a,b,x0,M)
% Metodo de Euler (Potenciales variables)para una variable
global lambda cx t
% Datos
%         - a y b son los extremos derecho e izquierdo del intervalo
%         - x0 es la condicion inicial y(a)
%         - M es el numero de pasos
%  Resultado
%         - E1 = [T' X'] siendo T el vector de las abscisas e X el vector
%           de las ordenadas

h =(b-a)/M;
lambda = 0.001;
```

```
T = zeros(1,M+1);
X = zeros(1,M+1);
T =a:h:b
X(1)=x0;

for j=1:M
    if j==1
        cx=x0;
    else
        cx = X(j);
    end
    t= T(j);
    [minimo,fval] = fminsearch(@phi2,cx);
    Min(j) = minimo;
    X(j+1) = X(j) - (h/lambda)*(X(j)-Min(j));
end

E1 = [T' X'];
```

### 6.3.2 EULER para dos variables

```
function [E1,E2] = eulerC2(a,b,x0,y0,M)
% Metodo de Euler (Potenciales variables) para dos variables.

global cx cy lambda t
% Datos
%         - a y b son los extremos derecho e izquierdo del intervalo
%         - x0 es la condicion inicial para la variable x
%         - y0 es la condicion inicial para la variable y
%         - M es el numero de pasos
%  Resultado
%         - E1 = [T' X'] siendo T el vector de las abscisas e X el vector
%           de las ordenadas
%         - E2 = [T' Y'] siendo T el vector de las abscisas e Y el vector
%         de las ordenadas.

h =(b-a)/M;
lambda = 0.01;
T = zeros(1,M+1);
X = zeros(1,M+1);
Y = zeros(1,M+1);
T = a:h:b;
X(1) = x0;
Y(1) = y0;

for j=1:M
```

```
    if j==1
        cx = x0;
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
    end
    t = T(j);
    [minimo,fval] = fminsearch(@phi2,[cx,cy]);
    X(j+1) = X(j) - (h/lambda)*(X(j)-minimo(1));
    Y(j+1) = Y(j) - (h/lambda)*(Y(j)-minimo(2));
end

E1 = [T' X'];
E2 = [T' Y'];
```

### 6.3.3 RUNGE KUTTA 4 para una variable

```
% Metodo Runge kutta 4 potenciales variables para una variable.
function [RK1]= rk4_C(a,b,x0,M)
global cx lambda minimox puntox t

h = (b-a)/M;
lambda = 0.001;
T = zeros(1,M+1);
T = a:h:b;
X = zeros(1,M+1);
X(1) = x0;

for j=1:M
    if j==1
        cx = x0;
    else
        cx = X(j);
    end

    t= T(j);
    puntox = cx;
    minimox = calculamin(puntox);
    k1 = feval('yo',puntox);
    t= T(j)+(h/2);
    puntox = cx + ((h*k1)/2);
    minimox = calculamin(puntox);
    k2 = feval('yo',puntox);
    t= T(j)+(h/2);
    puntox = cx + ((h*k2)/2);
```

```
    minimox = calculamin(puntox);
    k3 = feval('yo',puntox);
    t= T(j);
    puntox = cx + h*k3;
    minimox = calculamin(puntox);
    k4 = feval('yo',puntox);
    X(j+1)= X(j) + (h/6)*(k1+2*k2+2*k3+k4);

end

RK1=[T' X'];
```

### 6.3.4 RUNGE KUTTA 4 para dos variables

```
% Metodo Runge kutta 4 potenciales variables para dos variables.
function [RK1,RK2]= rk4_C2(a,b,x0,y0,M)

global cx cy lambda minimo puntox puntoy t

h = (b-a)/M;
lambda = 0.01;
T = zeros(1,M+1);
T = a:h:b
X = zeros(1,M+1);
Y = zeros(1,M+1);
X(1) = x0;
Y(1) = y0;

for j=1:M
    if j==1
        cx = x0;
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
    end

    t = T(j);
    puntox = cx;
    puntoy= cy;
    punto = [puntox,puntoy];
    minimo = calculamin2(punto);
    k1 = feval('yo2',punto);

    t= T(j)+(h/2);
    puntox = cx + ((h*k1(1))/2);
```

```
    puntoy = cy + ((h*k1(2))/2);
    punto = [puntox,puntoy];
    minimo = calculamin2(punto);
    k2 = feval('yo2',punto);

    t= T(j)+(h/2);
    puntox = cx + ((h*k2(1))/2);
    puntoy = cy + ((h*k2(2))/2);
    punto = [puntox,puntoy];
    minimo = calculamin2(punto);
    k3 = feval('yo2',punto);

    t= T(j)+ h;
    puntox = cx + h*k3(1);
    puntoy = cy + h*k3(2);
    punto = [puntox,puntoy];
    minimo = calculamin2(punto);
    k4 = feval('yo2',punto);

    X(j+1)= X(j) + (h/6)*(k1(1)+2*k2(1)+2*k3(1)+k4(1));
    Y(j+1)= Y(j) + (h/6)*(k1(2)+2*k2(2)+2*k3(2)+k4(2));
end

RK1 = [T' X'];
RK2 = [T' Y'];
```

## 6.4 Algoritmos para problemas perturbados

### 6.4.1 EULER para una variable

```
function E1 = eulerD(a,b,x0,M)
% Metodo de Euler (problemas perturbados) para una variable
global lambda cx
% Datos
%         - a y b son los extremos derecho e izquierdo del intervalo
%         - x0 es la condicion inicial y(a)
%         - M es el numero de pasos
%  Resultado
%         - E1 = [T' X'] siendo T el vector de las abscisas e X el vector
%           de las ordenadas

h =(b-a)/M;
lambda = 0.001;
T = zeros(1,M+1);
X = zeros(1,M+1);
```

```
Min = zeros(1,M+1);
T = a:h:b;
X(1)=x0;

for j=1:M
    if j==1
        cx = x0;
    else
        cx = X(j);
    end
    [minimo,fval] = fminsearch(@phi2,cx);
    Min(j) = minimo;
    X(j+1) = X(j) - (h/lambda)*(X(j)-Min(j))+ h*feval('F',X(j));
end

E1 = [T' X'];
```

### 6.4.2 EULER para dos variables

```
function [E1,E2] = eulerD2(a,b,x0,y0,M)
% Metodo de Euler (problemas perturbados) para dos variables.

global cx cy lambda
% Datos
%         - a y b son los extremos derecho e izquierdo del intervalo
%         - x0 es la condicion inicial para la variable x
%         - y0 es la condicion inicial para la variable y
%         - M es el numero de pasos
%  Resultado
%         - E1 = [T' X'] siendo T el vector de las abscisas e X el vector
%           de las ordenadas
%         - E2 = [T' Y'] siendo T el vector de las abscisas e Y el vector
%         de las ordenadas.

h =(b-a)/M;
lambda = 0.001;
T = zeros(1,M+1);
X = zeros(1,M+1);
Y = zeros(1,M+1);
T = a:h:b;
X(1) = x0;
Y(1) = y0;

for j=1:M
    if j==1
        cx = x0;
```

```
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
    end
    [minimo,fval] = fminsearch(@phi2,[cx,cy]);
    F2aux = F2([X(j),Y(j)]);
    X(j+1) = X(j) - (h/lambda)*(X(j)-minimo(1))+ h*F2aux(1);
    Y(j+1) = Y(j) - (h/lambda)*(Y(j)-minimo(2))+ h*F2aux(2);
end

E1 = [T' X'];
E2 = [T' Y'];
```

### 6.4.3 RUNGE KUTTA 4 para una variable

```
% Metodo de Runge kutta 4 para problemas perturbados de una variable
function [RK1]= rk4_D(a,b,x0,M)

global cx lambda minimox puntox

h = (b-a)/M;
lambda = 0.001;
T = zeros(1,M+1);
T = a:h:b;
X = zeros(1,M+1);
X(1) = x0;

for j=1:M
    if j==1
        cx = x0;
    else
        cx = X(j);
    end

    puntox = cx;
    minimox = calculamin(puntox)
    k1 = feval('yo',puntox)
    puntox = cx + ((h*k1)/2)
    minimox = calculamin(puntox)
    k2 = feval('yo',puntox)
    puntox = cx + ((h*k2)/2)
    minimox = calculamin(puntox)
    k3 = feval('yo',puntox)
    puntox = cx + h*k3
    minimox = calculamin(puntox)
```

```
    k4 = feval('yo',puntox)
    % calculo de los coeficientes del termino Fuente F.
    q1 = feval('F',cx);
    q2 = feval('F',cx + (h*q1)/2);
    q3 = feval('F',cx + (h*q2)/2);
    q4 = feval('F',cx + h*q3);
    % nueva actualizacion con un termino fuente.
    X(j+1)= X(j) + (h/6)*(k1+2*k2+2*k3+k4)+ (h/6)*(q1 + 2*q2 +2*q3 +q4);

end

RK1=[T' X'];
```

### 6.4.4 RUNGE KUTTA 4 para dos variables

```
% Metodo de Runge - kutta 4 para problemas perturbados de dos variables
function [RK1,RK2]= rk4_D2(a,b,x0,y0,M)
global cx cy lambda minimo puntox puntoy

h = (b-a)/M;
lambda = 0.001;

T = zeros(1,M+1);
T = a:h:b;
X = zeros(1,M+1);
Y = zeros(1,M+1);

X(1) = x0;
Y(1) = y0;

for j=1:M
    if j==1
        cx = x0;
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
    end

    puntox = cx;
    puntoy = cy;
    punto = [puntox,puntoy];
    minimo = calculamin2(punto);
    k1 = feval('yo2',punto);
    puntox = cx + ((h*k1(1))/2);
    puntoy = cy + ((h*k1(2))/2);
```

```
    punto = [puntox,puntoy];
    minimo = calculamin2(punto);
    k2 = feval('yo2',punto);
    puntox = cx + ((h*k2(1))/2);
    puntoy = cy + ((h*k2(2))/2);
    punto = [puntox,puntoy];
    minimo = calculamin2(punto);
    k3 = feval('yo2',punto);
    puntox = cx + h*k3(1);
    puntoy = cy + h*k3(2);
    punto = [puntox,puntoy];
    minimo = calculamin2(punto);
    k4 = feval('yo2',punto);

    % Cculo de los coeficientes del termino Fuente F.

    q1 = feval('F2',[cx,cy]);
    q2 = feval('F2',[cx + (h/2)*q1(1),cy + (h/2)*q1(2)]);
    q3 = feval('F2',[cx + (h/2)*q2(1),cy+ (h/2)*q2(2)]);
    q4 = feval('F2',[cx + h*q3(1),cy + h*q3(2)]);
    % Nueva actualizacion con un termino fuente.
    X(j+1)= X(j) + (h/6)*(k1(1)+2*k2(1)+2*k3(1)+k4(1))
          +(h/6)*(q1(1) + 2*q2(1) +2*q3(1) +q4(1));
    Y(j+1)= Y(j) + (h/6)*(k1(2)+2*k2(2)+2*k3(2)+k4(2))
          +(h/6)*(q1(2) + 2*q2(2) +2*q3(2) +q4(2));

end

RK1=[T' X'];
RK2=[T' Y'];
```

## 6.5 Algoritmos para potenciales variables y problemas perturbados

### 6.5.1 EULER para una variable

```
function E1 = eulerCD(a,b,x0,M)
% Metodo de Euler (potenciales variables) + (problemas perturbados) para una variable
global lambda cx t
% Datos
%         - a y b son los extremos derecho e izquierdo del intervalo
%         - x0 es la condicion inicial y(a)
%         - M es el numero de pasos
%  Resultado
%         - E1 = [T' X'] siendo T el vector de las abscisas e X el vector
%           de las ordenadas

h =(b-a)/M;
lambda = 0.001;
T = zeros(1,M+1);
X = zeros(1,M+1);
Min = zeros(1,M+1);
T = a:h:b;
X(1)=x0;

for j=1:M
    if j==1
        cx = x0;
    else
        cx = X(j);
    end
    t = T(j);
    [minimo,fval] = fminsearch(@phi2,cx);
    Min(j) = minimo;
    X(j+1) = X(j) - (h/lambda)*(X(j)-Min(j))+ h*feval('F',X(j));
end

E1 = [T' X'];
```

### 6.5.2 EULER para dos variables

```
function [E1,E2] = eulerCD2(a,b,x0,y0,M)
% Metodo de Euler (potenciales variables) +(problemas perturbados) para dos variables.

global cx cy lambda t
% Datos
%         - a y b son los extremos derecho e izquierdo del intervalo
```

```
%         - x0 es la condicion inicial y(a) para la variable x
%         - y0 es la condicion inicial para la variable y
%         - M es el numero de pasos
%  Resultado
%         - E1 = [T' X'] siendo T el vector de las abscisas e X el vector
%           de las ordenadas
%         - E2 = [T' Y'] siendo T el vector de las abscisas e Y el vector
%         de las ordenadas.

h =(b-a)/M;
lambda = 0.001;
T = zeros(1,M+1);
X = zeros(1,M+1);
Y = zeros(1,M+1);
T = a:h:b;
X(1) = x0;
Y(1) = y0;

for j=1:M
    if j==1
        cx = x0;
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
    end
    t = T(j);
    [minimo,fval] = fminsearch(@phi2,[cx,cy]);
    F2aux = F2([X(j),Y(j)]);
    X(j+1) = X(j) - (h/lambda)*(X(j)-minimo(1))+ h*F2aux(1);
    Y(j+1) = Y(j) - (h/lambda)*(Y(j)-minimo(2))+ h*F2aux(2);
end

E1 = [T' X'];
E2 = [T' Y'];
```

### 6.5.3 RUNGE KUTTA 4 para una variable

```
% Metodo de Runge kutta 4
%(potenciales variables y problemas perturbados) para una variable
function [RK1]= rk4_CD(a,b,x0,M)

global cx lambda minimox puntox t

h = (b-a)/M;
lambda = 0.001;
```

```
T = zeros(1,M+1);
T = a:h:b;
X = zeros(1,M+1);
X(1) = x0;

for j=1:M
    if j==1
        cx = x0;
    else
        cx = X(j);
    end

    t= T(j);
    puntox = cx;
    minimox = calculamin(puntox);
    k1 = feval('yo',puntox);
    q1 = feval('F',cx);
    t= T(j)+(h/2);
    puntox = cx + ((h*k1)/2);
    minimox = calculamin(puntox);
    k2 = feval('yo',puntox);
    q2 = feval('F',cx + (h/2)*q1);
    t= T(j)+(h/2);
    puntox = cx + ((h*k2)/2);
    minimox = calculamin(puntox);
    k3 = feval('yo',puntox);
    q3 = feval('F',cx + (h/2)*q2);
    t= T(j);
    puntox = cx + h*k3;
    minimox = calculamin(puntox);
    k4 = feval('yo',puntox);
    q4 = feval('F',cx + h*q3);

    X(j+1)= X(j) + (h/6)*(k1+2*k2+2*k3+k4)+ (h/6)*(q1 + 2*q2 +2*q3 +q4);

end

RK1=[T' X'];
```

### 6.5.4 RUNGE KUTTA 4 para dos variables

```
% Metodo de Runge kutta 4 potenciales variables y problemas perturbados para dos variables
function [RK1,RK2]= rk4_CD2(a,b,x0,y0,M)

global cx cy lambda minimo puntox puntoy t
```

```
h = (b-a)/M;
lambda = 0.001;
T = zeros(1,M+1);
T = a:h:b;
X = zeros(1,M+1);
Y = zeros(1,M+1);
X(1) = x0;
Y(1) = y0;

for j=1:M
    if j==1
        cx = x0;
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
    end

    t = T(j);
    puntox = cx;
    puntoy= cy;
    punto = [puntox,puntoy];
    minimo = calculamin2(punto);
    k1 = feval('yo2',punto);
    q1 = feval('F2',[cx,cy]);

    t= T(j)+(h/2);
    puntox = cx + ((h*k1(1))/2);
    puntoy = cy + ((h*k1(2))/2);
    punto = [puntox,puntoy];
    minimo = calculamin2(punto);
    k2 = feval('yo2',punto);
    q2 = feval('F2',[cx + (h/2)*q1(1),cy + (h/2)*q1(2)]);

    t= T(j)+(h/2);
    puntox = cx + ((h*k2(1))/2);
    puntoy = cy + ((h*k2(2))/2);
    punto = [puntox,puntoy];
    minimo = calculamin2(punto);
    k3 = feval('yo2',punto);
    q3 = feval('F2',[cx + (h/2)*q2(1),cy + (h/2)*q2(2)]);

    t= T(j)+ h;
    puntox = cx + h*k3(1);
    puntoy = cy + h*k3(2);
    punto = [puntox,puntoy];
```

```
    minimo = calculamin2(punto);
    k4 = feval('yo2',punto);
    q4 = feval('F2',[cx + h*q3(1),cy + h*q3(2)]);

    X(j+1)= X(j) + (h/6)*(k1(1)+2*k2(1)+2*k3(1)+k4(1))
          +(h/6)*(q1(1) + 2*q2(1) +2*q3(1) +q4(1));
    Y(j+1)= Y(j) + (h/6)*(k1(2)+2*k2(2)+2*k3(2)+k4(2))
          +(h/6)*(q1(2) + 2*q2(2) +2*q3(2) +q4(2));

end

RK1=[T' X'];
RK2=[T' Y'];
```

## 6.6 Algoritmos para problemas bipotenciales

### 6.6.1 EULER para una variable

```
% Metodo de Euler problemas bipotenciales para una variable.
function E1 = eulerE(a,b,x0,M)

global cx lambda1 lambda2 t
% Datos
%         - a y b son los extremos derecho e izquierdo del intervalo
%         - x0 es la condicion inicial de la variable X
%         - M es el numero de pasos
%  Resultado
%         - E1 = [T' X'] siendo T el vector de las abscisas e X el vector
%           de las ordenadas

h =(b-a)/M;
lambda1 = 0.001;
lambda2 = 0.001;
T = zeros(1,M+1);
X = zeros(1,M+1);
T = a:h:b;
X(1) = x0;

for j=1:M
    if j==1
        cx = x0;
    else
        cx = X(j);
    end
    t = T(j);
    beta = (1+t)^2;
    % Calculo de los minimos de las funciones f y g
    [minimof,fvalf] = fminsearch(@f,cx);
    [minimog,fvalg] = fminsearch(@g,cx);
    X(j+1) = X(j) - (h/lambda1)*(X(j)-minimof)- beta*(h/lambda2)*(X(j)-minimog);
end

E1 = [T' X'];
```

### 6.6.2 EULER para dos variables

```
% Metodo de Euler problemas bipotenciales para dos variables.
function [E1,E2] = eulerE2(a,b,x0,y0,M)
global cx cy lambda1 lambda2 t
% Datos
```

```
%         - a y b son los extremos derecho e izquierdo del intervalo
%         - x0 es la condicion inicial de la variable X
%         - y0 es la condicion inicial de la variable Y
%         - M es el numero de pasos
%  Resultado
%         - E1 = [T' X'] siendo T el vector de las abscisas e X el vector
%           de las ordenadas
%         - E2 = [T' Y'] siendo T el vector de las abscisas e Y el vector
%           de las ordenadas

h =(b-a)/M;
lambda1 = 0.001;
lambda2 = 0.01;
T = zeros(1,M+1);
X = zeros(1,M+1);
Y = zeros(1,M+1);
T = a:h:b;
X(1) = x0;
Y(1) = y0;

for j=1:M
    if j==1
        cx = x0;
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
    end
    t = T(j);
    beta= (1+t)^2;
    % Calculo de los minimos de las funciones f y g
    [minimof,fvalf] = fminsearch(@f,[cx,cy]);
    [minimog,fvalg] = fminsearch(@g,[cx,cy]);
    % Actualizaci de X e Y : obtencion de las soluciones
    X(j+1) = X(j) - (h/lambda1)*(X(j)-minimof(1))- beta*(h/lambda2)*(X(j)-minimog(1));
    Y(j+1) = Y(j) - (h/lambda1)*(Y(j)-minimof(2))- beta*(h/lambda2)*(Y(j)-minimog(2));
end

E1 = [T' X'];
E2 = [T' Y'];
```

### 6.6.3 RUNGE KUTTA 4 para una variable

```
% Metodo de Runge kutta 4 Problemas Bipotenciales para una variable.
function [RK1]= rk4_E(a,b,x0,M)
```

```
global cx lambda1 lambda2 minimof minimog puntoxf puntoxg t

h = (b-a)/M;
lambda1 = 0.001;
lambda2 =0.001;
T = zeros(1,M+1);
T = a:h:b;
X = zeros(1,M+1);
X(1) = x0;

for j=1:M
    if j==1
        cx = x0;
    else
        cx = X(j);
    end
    t = T(j);
    beta= (1+t)^2;
    puntoxf = cx;
    minimof = calculaminf(puntoxf);
    k1 = feval('yof',puntoxf);
    puntoxf = cx + ((h*k1)/2);
    minimof = calculaminf(puntoxf);
    k2 = feval('yof',puntoxf);
    puntoxf = cx + ((h*k2)/2);
    minimof = calculaminf(puntoxf);
    k3 = feval('yof',puntoxf);
    puntoxf = cx + h*k3;
    minimof = calculaminf(puntoxf);
    k4 = feval('yof',puntoxf);

    puntoxg = cx;
    minimog = calculaming(puntoxg);
    q1 = feval('yog',puntoxg);
    puntoxg = cx + ((h*q1)/2);
    minimog = calculaming(puntoxg);
    q2 = feval('yog',puntoxg);
    puntoxg = cx + ((h*q2)/2);
    minimog = calculaming(puntoxg);
    q3 = feval('yog',puntoxg);
    puntoxg = cx + h*q3;
    minimog = calculaming(puntoxg);
    q4 = feval('yog',puntoxg);

  X(j+1)= X(j) + (h/6)*(k1+2*k2+2*k3+k4)-beta*(h/6)*(q1 + 2*q2 + 2*q3 + q4);
```

```
end

RK1=[T' X'];
```

### 6.6.4 Función yof.m

```
function p = yof(x)
% Funcion Yoshida para la funcion f y para una variable
global lambda1 minimof
p = -(1/lambda1)*(x-minimof);
```

### 6.6.5 Función yog.m

```
function p = yog(x)
global lambda2 minimog
% Funcion Yoshida para la funcion g para una variable
p = (1/lambda2)*(x-minimog);
```

### 6.6.6 Función calculaminf.m

```
 function minimo = calculaminf(z)
 [min,fval] = fminsearch(@f,z);
 minimo = min;
```

### 6.6.7 Función calculaming.m

```
function minimo = calculaming(z)
 [min,fval] = fminsearch(@g,z);
 minimo = min;
```

### 6.6.8 RUNGE KUTTA 4 para dos variables

```
% Metodo de Runge kutta 4 Problemas Bipotenciales para dos variables.
function [RK1,RK2]= rk4_E2(a,b,x0,y0,M)

global cx cy lambda1 lambda2 minimof minimog puntoxf puntoyf puntoxg puntoyg t

h = (b-a)/M;
lambda1 = 0.001;
lambda2 = 0.01;
T = zeros(1,M+1);
T = a:h:b;
X = zeros(1,M+1);
Y = zeros(1,M+1);
X(1) = x0;
Y(1) = y0;
```

```
for j=1:M
    if j==1
        cx = x0;
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
    end
    t = T(j);
    beta = (1+t)^2;
    puntoxf = cx;
    puntoyf = cy;
    puntoxg = cx;
    puntoyg = cy;
    puntof = [puntoxf,puntoyf];
    puntog = [puntoxg,puntoyg];
    minimof = calculaminf(puntof);
    minimog = calculaming(puntog);

    k1 = feval('yo2f',puntof);
    q1 = feval('yo2g',puntog);

    puntoxf = cx + ((h*k1(1))/2);
    puntoyf = cy + ((h*k1(2))/2);
    puntoxg = cx + ((h*q1(1))/2);
    puntoyg = cy + ((h*q1(2))/2);
    puntof = [puntoxf,puntoyf];
    puntog = [puntoxg,puntoyg];
    minimof = calculaminf(puntof);
    minimog = calculaming(puntog);

    k2 = feval('yo2f',puntof);
    q2 = feval('yo2g',puntog);

    puntoxf = cx + ((h*k2(1))/2);
    puntoyf = cy + ((h*k2(2))/2);
    puntoxg = cx + ((h*q2(1))/2);
    puntoyg = cy + ((h*q2(2))/2);
    puntof = [puntoxf,puntoyf];
    puntog = [puntoxg,puntoyg];
    minimof = calculaminf(puntof);
    minimog = calculaming(puntog);

    k3 = feval('yo2f',puntof);
    q3 = feval('yo2g',puntog);
```

```
    puntoxf = cx + h*k3(1);
    puntoyf = cy + h*k3(2);
    puntoxg = cx + h*q3(1);
    puntoyg = cy + h*q3(2);
    puntof = [puntoxf,puntoyf];
    puntog = [puntoxg,puntoyg];
    minimof = calculaminf(puntof);
    minimog = calculaming(puntog);

    k4 = feval('yo2f',puntof);
    q4 = feval('yo2g',puntog);

  X(j+1)= X(j) - (h/(6*lambda1))*(k1(1)+2*k2(1)+2*k3(1)+k4(1))
        -beta*(h/(6*lambda2))*(q1(1) + 2*q2(1) + 2*q3(1) + q4(1));
  Y(j+1)= Y(j) - (h/(6*lambda1))*(k1(2)+2*k2(2)+2*k3(2)+k4(2))
        -beta*(h/(6*lambda2))*(q1(2) + 2*q2(2) + 2*q3(2) + q4(2));

end

RK1 = [T' X'];
RK2 = [T' Y'];
```

### 6.6.9 Función yo2f.m

```
function p = yo2f(x)
global minimof
% Funcion Yoshida para la funcion f para dos variables.
yo_1 = x(1)-minimof(1);
yo_2 = x(2)-minimof(2);
p = [yo_1,yo_2];
```

### 6.6.10 Función yo2g.m

```
function p = yo2g(x)
global minimog
% Funcion Yoshida para la funcion g para dos variables
yo_1 = x(1)-minimog(1);
yo_2 = x(2)-minimog(2);
p = [yo_1,yo_2];
```

**Nota:** Cabe destacar que para el caso de la definición de los ficheros de las funciones phi2.m, en Euler se hace uso de los parámetros `cx` y `cy`, uno para cada variable; y en cambio para Runge Kutta se hace uso de `puntox` y `puntoy` (en algunos casos se hace uso de los parámetros `puntoxf`,`puntoyf`, `puntoxg`, `puntoyg`) según el tipo de problema que estemos tratando.

# Capítulo 7

# Anexos SolvOpt

## 7.1 Algoritmos para potenciales variables con SolvOpt

### 7.1.1 EULER

```
function [E1,E2] = eulerSO2(a,b,x0,y0,M)
% Metodo de Euler (Potenciales variables) para dos variables utilizando
% SolvOpt

global cx cy lambda t g1 g2
% Datos
%         - a y b son los extremos derecho e izquierdo del intervalo
%         - x0 es la condicion inicial para la variable x
%         - y0 es la condicion inicial para la variable y
%         - M es el numero de pasos
%  Resultado
%         - E1 = [T' X'] siendo T el vector de las abscisas e X el vector
%           de las ordenadas
%         - E2 = [T' Y'] siendo T el vector de las abscisas e Y el vector
%         de las ordenadas.

h =(b-a)/M;
lambda = 0.001;
T = zeros(1,M+1);
X = zeros(1,M+1);
Y = zeros(1,M+1);
T = a:h:b;
X(1) = x0;
Y(1) = y0;
```

```
for j=1:M
    if j==1
        cx = x0;
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
    end
    t = T(j);
    % Ejemplo 1
%     g1 = cos(t);
%     g2 = sin(t);
    % Ejemplo 2
    g1 = cos(t);
    g2 = sin(t);
    vector=[X(1),Y(1),g1,g2];
    [mini,f] = solvopt(vector,'funobjetivo',[],[],'funcMr',[]);
    minimo = [mini(:,1),mini(:,2)];

    X(j+1) = X(j) - (h/lambda)*(X(j)-minimo(1));
    Y(j+1) = Y(j) - (h/lambda)*(Y(j)-minimo(2));
end

E1 = [T' X'];
E2 = [T' Y'];
```

### 7.1.2 Función objetivo para Euler

```
function n = funobjetivo(z)
global lambda cx cy
n =  sqrt(((z(1)-z(3))^2 + (z(2)-z(4))^2))
  + (1/(2*lambda))*((z(1)-cx)^2 + (z(2)-cy)^2);
```

### 7.1.3 RUNGE KUTTA

```
% Runge kutta 4 para dos variables.Potenciales Variables
function [RK1,RK2]= rk4_SO2(a,b,x0,y0,M)

global cx cy lambda minimo puntox puntoy t g1 g2

h = (b-a)/M;
lambda = 0.001;
T = zeros(1,M+1);
T = a:h:b;
X = zeros(1,M+1);
Y = zeros(1,M+1);
```

```
X(1) = x0;
Y(1) = y0;

for j=1:M
    if j==1
        cx = x0;
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
    end

    t = T(j);
     % Ejemplo 1
    g1 = cos(t);
    g2 = sin(t);
    % Ejemplo 2
%     g1 = cos(t);
%     g2 = 1+t;
    puntox = cx;
    puntoy= cy;
    punto = [puntox,puntoy];
    vector=[x0,y0,g1,g2];
    minimo = calculaminSO2(vector);
    k1 = feval('yoSO2',punto);

    t= T(j)+(h/2);
    puntox = cx + ((h*k1(1))/2);
    puntoy = cy + ((h*k1(2))/2);
    punto = [puntox,puntoy];
    vector=[x0,y0,g1,g2];
    minimo = calculaminSO2(vector);
    k2 = feval('yoSO2',punto);

    t= T(j)+(h/2);
    puntox = cx + ((h*k2(1))/2);
    puntoy = cy + ((h*k2(2))/2);
    punto = [puntox,puntoy];
    vector=[x0,y0,g1,g2];
    minimo = calculaminSO2(vector);
    k3 = feval('yoSO2',punto);

    t= T(j)+ h;
    puntox = cx + h*k3(1);
    puntoy = cy + h*k3(2);
    punto = [puntox,puntoy];
```

```
    vector=[x0,y0,g1,g2];
    minimo = calculaminSO2(vector);
    k4 = feval('yoSO2',punto);

    X(j+1)= X(j) + (h/6)*(k1(1)+2*k2(1)+2*k3(1)+k4(1));
    Y(j+1)= Y(j) + (h/6)*(k1(2)+2*k2(2)+2*k3(2)+k4(2));
end

RK1 = [T' X'];
RK2 = [T' Y'];
```

### 7.1.4 Función objetivo para Runge Kutta

```
function n = funobjetivo(z)
global lambda puntox puntoy
n =  sqrt(((z(1)-z(3))^2 + (z(2)-z(4))^2))
  + (1/(2*lambda))*((z(1)-puntox)^2 + (z(2)-puntoy)^2);
```

### 7.1.5 Función calculaminSO2.m

```
function [mini] = calculaminSO2(k)
[mini,f] = solvopt(k,'funobjetivo',[],[],'funcMr',[]);
minimo = [mini(:,1),mini(:,2)];
```

## 7.2 Algoritmos para problemas perturbados con SolvOpt

### 7.2.1 EULER

```
function [E1,E2] = eulerSOD2(a,b,x0,y0,M)
% Metodo de Euler (problemas perturbados) para dos variables utilizando SolvOpt.

global cx cy lambda g1 g2
% Datos
%         - a y b son los extremos derecho e izquierdo del intervalo
%         - x0 es la condicion inicial y(a) para la variable x
%         - y0 es la condicion inicial para la variable y
%         - M es el numero de pasos
%  Resultado
%         - E1 = [T' X'] siendo T el vector de las abscisas e X el vector
%           de las ordenadas
%         - E2 = [T' Y'] siendo T el vector de las abscisas e Y el vector
%         de las ordenadas.

h =(b-a)/M;
```

```
lambda = 0.001;
T = zeros(1,M+1);
X = zeros(1,M+1);
Y = zeros(1,M+1);
T = a:h:b;
X(1) = x0;
Y(1) = y0;

for j=1:M
    if j==1
        cx = x0;
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
    end
     % Ejemplo 1
    g1 = 3;
    g2 = 1;
    % Ejemplo 2
%     g1 = 1;
%     g2 = 2;

    vector=[X(1),Y(1),g1,g2];
    [mini,f] = solvopt(vector,'funobjetivo',[],[],'funcMr',[])
    minimo = [mini(:,1),mini(:,2)];
    F2aux = FSOD2([X(j),Y(j)])

    X(j+1) = X(j) - (h/lambda)*(X(j)-minimo(1))+ h*F2aux(1);
    Y(j+1) = Y(j) - (h/lambda)*(Y(j)-minimo(2))+ h*F2aux(2);
end

E1 = [T' X'];
E2 = [T' Y'];
```

### 7.2.2 RUNGE KUTTA

```
% Metodo Runge kutta 4 (Problemas perturbados) para dos variables utilizando SolvOpt
function [RK1,RK2]= rk4_SOD2(a,b,x0,y0,M)

global cx cy lambda minimo puntox puntoy g1 g2
h = (b-a)/M;
lambda = 0.001;
T = zeros(1,M+1);
T = a:h:b;
X = zeros(1,M+1);
```

```
Y = zeros(1,M+1);
X(1) = x0;
Y(1) = y0;

for j=1:M
    if j==1
        cx = x0;
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
    end

   g1 = 3;
   g2 = 1;

    puntox = cx;
    puntoy = cy;
    punto = [puntox,puntoy];
    vector=[x0,y0,g1,g2];
    minimo = calculaminSOD2(vector);
    k1 = feval('yoSOD2',punto);
    puntox = cx + ((h*k1(1))/2);
    puntoy = cy + ((h*k1(2))/2);
    punto = [puntox,puntoy];
    vector=[x0,y0,g1,g2];
    minimo = calculaminSOD2(vector);
    k2 = feval('yoSOD2',punto);
    puntox = cx + ((h*k2(1))/2);
    puntoy = cy + ((h*k2(2))/2);
    punto = [puntox,puntoy];
    vector=[x0,y0,g1,g2];
    minimo = calculaminSOD2(vector);
    k3 = feval('yoSOD2',punto);
    puntox = cx + h*k3(1);
    puntoy = cy + h*k3(2);
    punto = [puntox,puntoy];
    vector=[x0,y0,g1,g2];
    minimo = calculaminSOD2(vector);
    k4 = feval('yoSOD2',punto);

    q1 = feval('FSOD2',[cx,cy]);
    q2 = feval('FSOD2',[cx + (h/2)*q1(1),cy + (h/2)*q1(2)]);
    q3 = feval('FSOD2',[cx + (h/2)*q2(1),cy+ (h/2)*q2(2)]);
    q4 = feval('FSOD2',[cx + h*q3(1),cy + h*q3(2)]);
    % nueva actualizacion con un termino fuente.
```

```
    X(j+1)= X(j) + (h/6)*(k1(1)+2*k2(1)+2*k3(1)+k4(1))
          +(h/6)*(q1(1) + 2*q2(1) +2*q3(1) +q4(1));
    Y(j+1)= Y(j) + (h/6)*(k1(2)+2*k2(2)+2*k3(2)+k4(2))
          +(h/6)*(q1(2) + 2*q2(2) +2*q3(2) +q4(2));

end

RK1=[T' X'];
RK2=[T' Y'];
```

## 7.3 Algoritmos para potenciales variables y problemas perturbados con SolvOpt

### 7.3.1 EULER

```
function [E1,E2] = eulerSOCD2(a,b,x0,y0,M)
% Metodo de Euler (potenciales variables) +(problemas perturbados) para dos
% variables utilizando SolvOpt.

global cx cy lambda t g1 g2
% Datos
%         - a y b son los extremos derecho e izquierdo del intervalo
%         - x0 es la condicion inicial y(a) para la variable x
%         - y0 es la condicion inicial para la variable y
%         - M es el numero de pasos
%  Resultado
%         - E1 = [T' X'] siendo T el vector de las abscisas e X el vector
%           de las ordenadas
%         - E2 = [T' Y'] siendo T el vector de las abscisas e Y el vector
%         de las ordenadas.

h =(b-a)/M;
lambda = 0.001;
T = zeros(1,M+1);
X = zeros(1,M+1);
Y = zeros(1,M+1);
T = a:h:b;
X(1) = x0;
Y(1) = y0;

for j=1:M
    if j==1
        cx = x0;
        cy = y0;
    else
```

```
        cx = X(j);
        cy = Y(j);
    end
    t = T(j);
    % Ejemplo 1 (circulo)
%     g1 = cos(t);
%     g2 = sin(t);
    % Ejemplo 1 (elipse)
    g1 = 2+ cos(t);
    g2 = 1;
    vector=[X(1),Y(1),g1,g2];
    [mini,f] = solvopt(vector,'funobjetivo',[],[],'funcMr',[]);
    minimo = [mini(:,1),mini(:,2)];
    F2aux = FSOCD2([X(j),Y(j)]);
    X(j+1) = X(j) - (h/lambda)*(X(j)-minimo(1))+ h*F2aux(1);
    Y(j+1) = Y(j) - (h/lambda)*(Y(j)-minimo(2))+ h*F2aux(2);
end

E1 = [T' X'];
E2 = [T' Y'];
```

### 7.3.2 RUNGE KUTTA

```
% Metodo Runge kutta 4 (potenciales variables) +(problemas perturbados) para dos
% variables utilizando SolvOpt.
function [RK1,RK2]= rk4_SOCD2(a,b,x0,y0,M)

global cx cy lambda minimo puntox puntoy t g1 g2

h = (b-a)/M;
lambda = 0.001;
T = zeros(1,M+1);
T = a:h:b;
X = zeros(1,M+1);
Y = zeros(1,M+1);
X(1) = x0;
Y(1) = y0;

for j=1:M
    if j==1
        cx = x0;
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
    end
```

```
    t = T(j);
    % Ejemplo 1 (circulo)
%     g1 = cos(t);
%     g2 = sin(t);
    % Ejemplo 1 (elipse)
    g1 = 2+ cos(t);
    g2 = 1;
    puntox = cx;
    puntoy= cy;
    punto = [puntox,puntoy];
    vector=[x0,y0,g1,g2];
    minimo = calculaminSOCD2(vector);
    k1 = feval('yoSOCD2',punto);
    q1 = feval('FSOCD2',[cx,cy]);

    t= T(j)+(h/2);
    puntox = cx + ((h*k1(1))/2);
    puntoy = cy + ((h*k1(2))/2);
    punto = [puntox,puntoy];
    vector=[x0,y0,g1,g2];
    minimo = calculaminSOCD2(vector);
    k2 = feval('yoSOCD2',punto);
    q2 = feval('FSOCD2',[cx + (h/2)*q1(1),cy + (h/2)*q1(2)]);

    t= T(j)+(h/2);
    puntox = cx + ((h*k2(1))/2);
    puntoy = cy + ((h*k2(2))/2);
    punto = [puntox,puntoy];
    vector=[x0,y0,g1,g2];
    minimo = calculaminSOCD2(vector);
    k3 = feval('yoSOCD2',punto);
    q3 = feval('FSOCD2',[cx + (h/2)*q2(1),cy + (h/2)*q2(2)]);

    t= T(j)+ h;
    puntox = cx + h*k3(1);
    puntoy = cy + h*k3(2);
    punto = [puntox,puntoy];
    vector=[x0,y0,g1,g2];
    minimo = calculaminSOCD2(vector);
    k4 = feval('yoSOCD2',punto);
    q4 = feval('FSOCD2',[cx + h*q3(1),cy + h*q3(2)]);

    X(j+1)= X(j) + (h/6)*(k1(1)+2*k2(1)+2*k3(1)+k4(1)
          +(h/6)*(q1(1) + 2*q2(1) +2*q3(1) +q4(1));
    Y(j+1)= Y(j) + (h/6)*(k1(2)+2*k2(2)+2*k3(2)+k4(2))
```

```
          +(h/6)*(q1(2) + 2*q2(2) +2*q3(2) +q4(2));

end

RK1=[T' X'];
RK2=[T' Y'];
```

## 7.4 Algoritmos para problemas bipotenciales con SolvOpt

### 7.4.1 EULER

```
function [E1,E2] = eulerESO2(a,b,x0,y0,M)
% Metodo de Euler Problemas Bipotenciales para dos variables utilizando
% SolvOpt

global cx cy lambda1 lambda2 t g1 g2 f1 f2
%Datos
%         - a y b son los extremos derecho e izquierdo del intervalo
%         - x0 es la condicion inicial para la variable x
%         - y0 es la condicion inicial para la variable y
%         - M es el numero de pasos
%  Resultado
%         - E1 = [T' X'] siendo T el vector de las abscisas e X el vector
%           de las ordenadas
%         - E2 = [T' Y'] siendo T el vector de las abscisas e Y el vector
%         de las ordenadas.

h =(b-a)/M;
lambda1 = 0.001;
lambda2 = 0.001;
T = zeros(1,M+1);
X = zeros(1,M+1);
Y = zeros(1,M+1);
T = a:h:b;
X(1) = x0;
Y(1) = y0;

for j=1:M
    if j==1
        cx = x0;
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
```

```
    end
    t = T(j);
    beta= (1+t)^2;
    f1 = 0;
    f2 = 0;
    g1= 3;
    g2 = 0;
%     f1 = 1;
%     f2 = 2;
%     g1 = sin(t);
%     g2 = 1+t;
    vector=[X(1),Y(1),f1,f2];
    [minif,fn] = solvopt(vector,'funobjetivoF',[],[],'funcMrF',[]);
    minimof = [minif(:,1),minif(:,2)];
    vectorb=[X(1),Y(1),g1,g2];
    [minig,gn] = solvopt(vectorb,'funobjetivoG',[],[],'funcMrG',[]);
    minimog = [minig(:,1),minig(:,2)];

    X(j+1) = X(j) - (h/lambda1)*(X(j)-minimof(1))- beta*(h/lambda2)*(X(j)-minimog(1));
    Y(j+1) = Y(j) - (h/lambda1)*(Y(j)-minimof(2))- beta*(h/lambda2)*(Y(j)-minimog(2));
end

E1 = [T' X'];
E2 = [T' Y'];
```

### 7.4.2 Función objetivo primer potencial para Euler

```
function n = funobjetivoF(z)
global lambda1 cx cy
n =  sqrt(((z(1)-z(3))^2 + (z(2)-z(4))^2)) + (1/(2*lambda1))*((z(1)-cx)^2 + (z(2)-cy)^2);
```

### 7.4.3 Función objetivo segundo potencial para Euler

```
function n = funobjetivoG(z)
global lambda2 cx cy
n =  sqrt(((z(1)-z(3))^2 + (z(2)-z(4))^2)) + (1/(2*lambda2))*((z(1)-cx)^2 + (z(2)-cy)^2);
```

### 7.4.4 RUNGE KUTTA

```
% Modo de Runge kutta 4 Problemas Bipotenciales para dos variables
% utilizando SolvOpt
function [RK1,RK2]= rk4_E2(a,b,x0,y0,M)

global cx cy lambda1 lambda2 minimof minimog puntoxf puntoyf puntoxg puntoyg t f1 f2 g1 g2

h = (b-a)/M;
```

```
lambda1 = 0.001;
lambda2 =0.001;
T = zeros(1,M+1);
T = a:h:b;
X = zeros(1,M+1);
Y = zeros(1,M+1);
X(1) = x0;
Y(1) = y0;

for j=1:M
    if j==1
        cx = x0;
        cy = y0;
    else
        cx = X(j);
        cy = Y(j);
    end
    t = T(j);
    beta = (1+t)^2;
    f1 = 0;
    f2 = 0;
    g1= 3;
    g2 = 0;

    puntoxf = cx;
    puntoyf = cy;
    puntoxg = cx;
    puntoyg = cy;
    puntof = [puntoxf,puntoyf,f1,f2];
    puntog = [puntoxg,puntoyg,g1,g2];
    minimof = calculaminf(puntof);
    minimog = calculaming(puntog);

    k1 = feval('yo2f',puntof);
    q1 = feval('yo2g',puntog);

    puntoxf = cx + ((h*k1(1))/2);
    puntoyf = cy + ((h*k1(2))/2);
    puntoxg = cx + ((h*q1(1))/2);
    puntoyg = cy + ((h*q1(2))/2);
    puntof = [puntoxf,puntoyf,f1,f2];
    puntog = [puntoxg,puntoyg,g1,g2];
    minimof = calculaminf(puntof);
    minimog = calculaming(puntog);

    k2 = feval('yo2f',puntof);
```

```
    q2 = feval('yo2g',puntog);

    puntoxf = cx + ((h*k2(1))/2);
    puntoyf = cy + ((h*k2(2))/2);
    puntoxg = cx + ((h*q2(1))/2);
    puntoyg = cy + ((h*q2(2))/2);
    puntof = [puntoxf,puntoyf,f1,f2];
    puntog = [puntoxg,puntoyg,g1,g2];
    minimof = calculaminf(puntof);
    minimog = calculaming(puntog);

    k3 = feval('yo2f',puntof);
    q3 = feval('yo2g',puntog);

    puntoxf = cx + h*k3(1);
    puntoyf = cy + h*k3(2);
    puntoxg = cx + h*q3(1);
    puntoyg = cy + h*q3(2);
    puntof = [puntoxf,puntoyf,f1,f2];
    puntog = [puntoxg,puntoyg,g1,g2];
    minimof = calculaminf(puntof);
    minimog = calculaming(puntog);

    k4 = feval('yo2f',puntof);
    q4 = feval('yo2g',puntog);

  X(j+1)= X(j) - (h/(6*lambda1))*(k1(1)+2*k2(1)+2*k3(1)+k4(1))
        -beta*(h/(6*lambda2))*(q1(1) + 2*q2(1) + 2*q3(1) + q4(1));
  Y(j+1)= Y(j) - (h/(6*lambda1))*(k1(2)+2*k2(2)+2*k3(2)+k4(2))
        -beta*(h/(6*lambda2))*(q1(2) + 2*q2(2) + 2*q3(2) + q4(2));

end

RK1 = [T' X'];
RK2 = [T' Y'];
```

### 7.4.5 Función objetivo primer potencial para Runge Kutta

```
function n = funobjF(z)
global lambda1 puntoxf puntoyf
n =  sqrt(((z(1)-z(3))^2 + (z(2)-z(4))^2)) +
 (1/(2*lambda1))*((z(1)-puntoxf)^2 + (z(2)-puntoyf)^2);
```

### 7.4.6 Función objetivo segundo potencial para Runge Kutta

```
function n = funobjG(z)
```

```
global lambda2 puntoxg puntoyg
n =  sqrt(((z(1)-z(3))^2 + (z(2)-z(4))^2)) +
 (1/(2*lambda2))*((z(1)-puntoxg)^2 + (z(2)-puntoyg)^2);
```

### 7.4.7 Función calculaminf.m

```
function minimof = calculaminf(z)
[minif,fn] = solvopt(z,'funobjF',[],[],'funcMrF',[]);
minimof = [minif(:,1),minif(:,2)];
```

### 7.4.8 Función calculaming.m

```
function minimog = calculaming(z)
[minig,gn] = solvopt(z,'funobjG',[],[],'funcMrG',[]);
minimog = [minig(:,1),minig(:,2)];
```

# 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] H. Brézis, Análisis Funcional, Alianza, 1984.

[6] F.H. Clarke, Optimization and Nonsmooth Analysis, 1990 (reimpresión de la edición de Wiley, 1983).

[7] A. Dontchev y F .Lempio, *Difference methods for differential inclusions: a survey*, SIAM Review, **34** (1992), 263–294.

[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] J.H. Mathew y K.D. Fink, Métodos numéricos con MATLAB, Prentice Hall, 2000.

[14] M.A. Raupp y O.R. Baiocchi, *Analysis of a RCL circuit in the dielectric breakdown limit*, Bol. Soc. Bras. Mat., **11** (1980), 241–258.

[15] M.A. Raupp y O.R. Baiocchi, *Analysis of a nonlinear series RCL circuit*, Revista Bras. Fis., bf 11 (1981), 211–230.

[16] R.T. Rockafellar, Convex Analysis, Princeton University Press, 1997 (1970, 1a. edición).

[17] R.T. Rockafellar y R. J.-B. Wets, Variational Analysis, Springer, 1998.