Análisis y Simulación de un Problema de Control Óptimo Regido por una Ecuación de Poisson

Javier Gatica, Vicente Opazo

Introducción

Este informe es una presentación didáctica del desarrollo mostrado en el libro *Optimal Control of Partial Differential Equations* [1] para el problema de control óptimo con una PDE como restricción. Imaginemos que queremos calentar una habitación aplicando calor en su interior de forma controlada. No solo nos interesa que la temperatura final se aproxime lo más posible a un valor deseado, sino también que el uso de energía sea lo más eficiente posible. Este tipo de situaciones se presenta con frecuencia en la ingeniería, la medicina y otros campos donde se busca influir sobre un sistema físico para llevarlo a un estado deseado, sin incurrir en un costo excesivo.

Desde el punto de vista matemático, esto conduce a los llamados *problemas de control óptimo*, en los que se desea determinar cómo actuar sobre un sistema para minimizar una cierta medida de error respecto a un objetivo, sumada al esfuerzo necesario para lograrlo. Como estos dos objetivos (precisión y economía de recursos) pueden ser contradictorios, se introduce un término regularizador que equilibra ambos intereses.

En este proyecto estudiamos un caso simple pero fundamental: controlar el estado de un sistema gobernado por la ecuación de Poisson, eligiendo apropiadamente el término fuente que aparece en la ecuación. Este modelo captura de forma idealizada el problema descrito, y nos permite analizar en profundidad cómo la matemática puede guiar decisiones óptimas sobre sistemas físicos distribuidos.

Formulación del Problema

Buscamos resolver el siguiente problema de control óptimo: determinar una fuente μ tal que la solución u del problema de Dirichlet para la ecuación de Poisson se aproxime a una función objetivo u_Ω, minimizando al mismo tiempo un costo cuadrático asociado al uso de la fuente. El problema se plantea como:

$$\begin{aligned} \min_{y,u} \quad & \frac{1}{2}\|y - z_d\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2(\Omega)} \\ \text{s.a.} \quad & -\Delta y = u \quad \text{en } \Omega, \\ & u = 0 \quad \text{en } \partial\Omega. \end{aligned}$$

A continuación, describimos con precisión los objetos matemáticos involucrados en esta formulación del problema:

- $\Omega \subset \mathbb{R}^d$ denota un dominio abierto, acotado y con frontera Lipschitz, sobre el cual se define el problema. Nos enfocaremos principalmente en el caso $d = 2$.
- $y : \Omega \to \mathbb{R}$ es la *variable de estado*, que representa, por ejemplo, la temperatura en el dominio. Es la solución de una ecuación de Poisson con fuente u, la cual llamaremos *ecuación de estado*.
- $u : \Omega \to \mathbb{R}$ es la *función de control*, que actúa como término fuente en la ecuación elíptica. Es la variable que se puede modificar para influir en el comportamiento del sistema.

- $z_d \in L^2(\Omega)$ es la *función objetivo* o *estado deseado*, que representa la distribución de temperatura que se desea alcanzar en el dominio.
- $\beta > 0$ es un parámetro de penalización que controla el equilibrio entre la cercanía al objetivo y el costo asociado al uso del control. Su valor regula la influencia del segundo término en la función objetivo.
- $L^2(\Omega)$ es el espacio de funciones cuadrado-integrables sobre Ω, donde vive el control μ y el objetivo u_Ω.
- $H_0^1(\Omega)$ es el espacio de funciones en $L^2(\Omega)$ con derivadas débiles de primer orden en $L^2(\Omega)$ y con traza nula en la frontera, donde se busca la solución u.
- La norma $\|\cdot\|_{L^2(\Omega)}$ denota la norma usual en el espacio de funciones cuadrado-integrables, es decir,

$$\|f\|_{L^2(\Omega)} := \left(\int_\Omega |f(x)|^2\, dx\right)^{1/2}.$$

- La norma en el espacio $H_0^1(\Omega)$ se define como

$$\|f\|_{H_0^1(\Omega)} := \|f\|_{L^2(\Omega)} + \|\nabla f\|_{L^2(\Omega)}$$

En resumen, el problema consiste en encontrar una pareja $(y, u) \in H_0^1(\Omega) \times L^2(\Omega)$ tal que y resuelva la ecuación de Poisson con fuente u y se minimice el funcional de costo compuesto por la desviación cuadrática respecto al objetivo y una penalización al uso del control.

Sistema Óptimo: Condiciones de Optimalidad

En esta sección nos proponemos estudiar con mayor profundidad el problema de control formulado anteriormente. Nuestro objetivo es, por un lado, demostrar que el problema está bien planteado, y por otro, caracterizar dicha solución mediante un sistema de ecuaciones que describa las condiciones de optimalidad de primer orden.

Para ello, haremos uso del *método adjunto*, una herramienta fundamental en el análisis de problemas de control óptimo, que permite transformar la minimización de un funcional sujeto a una restricción en un sistema acoplado de ecuaciones variacionales que involucra el estado, el control y una nueva variable conocida como *variable adjunta*.

Ecuación de Estado

Antes de analizar las condiciones de optimalidad del problema de control, es necesario estudiar la ecuación de estado que describe la evolución del sistema bajo la acción del control. En nuestro caso, la variable de estado y está gobernada por la ecuación de Poisson con condición de Dirichlet homogénea:

$$-\Delta y = u \quad \text{en } \Omega, \qquad y = 0 \quad \text{en } \partial\Omega.$$

Para analizar esta ecuación y garantizar su buena formulación (well-posedness), procedemos a deducir su formulación débil. Esta consiste en multiplicar la ecuación por una función de prueba $v \in H_0^1(\Omega)$, e integrar por partes el término de segundo orden utilizando que $y = 0$ en el borde. Obtenemos así:

$$\int_\Omega \nabla y \cdot \nabla v \, dx = \int_\Omega uv \, dx, \quad \forall v \in H_0^1(\Omega).$$

Definiendo la forma bilineal $a(y,v) := \int_\Omega \nabla y \cdot \nabla v \, dx$ y el funcional lineal $F(v) := \int_\Omega uv \, dx$, el problema se puede reescribir como: Encontrar $y \in H_0^1(\Omega)$ tal que

$$a(y,v) = F(v) \quad \forall v \in H_0^1(\Omega).$$

Con el fin de garantizar la existencia y unicidad de solución, recurrimos al Teorema de Lax-Milgram, el cual establece condiciones suficientes para la resolución de problemas variacionales de esta forma. Para ello, verificamos que las hipótesis del teorema se cumplen:

1. **Continuidad de la forma bilineal:** La forma bilineal $a(y,v) = \int_\Omega \nabla y \cdot \nabla v \, dx$ es continua sobre $H_0^1(\Omega)$. En efecto, por la desigualdad de Cauchy-Schwarz,

$$|a(y,v)| = \left| \int_\Omega \nabla y \cdot \nabla v \, dx \right| \leq \|\nabla y\|_{L^2(\Omega)} \|\nabla v\|_{L^2(\Omega)} \leq \|y\|_{H_0^1(\Omega)} \|v\|_{H_0^1(\Omega)},$$

   ya que $\|\nabla y\|_{L^2(\Omega)} \leq \|y\|_{H_0^1(\Omega)}$. Por lo tanto, $a(\cdot,\cdot)$ es continua con constante $C = 1$.

2. **Coercividad de la forma bilineal:** Para todo $y \in H_0^1(\Omega)$,

$$a(y,y) = \int_\Omega |\nabla y|^2 \, dx = \|\nabla y\|_{L^2(\Omega)}^2.$$

   Aplicando la desigualdad de Poincaré, existe una constante $C_P > 0$ tal que

$$\|y\|_{L^2(\Omega)} \leq C_P \|\nabla y\|_{L^2(\Omega)},$$

   lo que implica

$$\|y\|_{H_0^1(\Omega)}^2 = \|y\|_{L^2(\Omega)}^2 + \|\nabla y\|_{L^2(\Omega)}^2 \leq (1 + C_P^2) \|\nabla y\|_{L^2(\Omega)}^2 = (1 + C_P^2) a(y,y).$$

   Por tanto,

$$a(y,y) \geq \frac{1}{1 + C_P^2} \|y\|_{H_0^1(\Omega)}^2,$$

   y se satisface la coercividad con constante $\gamma = \frac{1}{1+C_P^2} > 0$.

3. **Continuidad del funcional lineal:** El funcional $F(v) = \int_\Omega uv \, dx$ es continuo sobre $H_0^1(\Omega)$ para todo $u \in L^2(\Omega)$. Por la desigualdad de Cauchy-Schwarz,

$$|F(v)| \leq \|u\|_{L^2(\Omega)} \|v\|_{L^2(\Omega)} \leq \|u\|_{L^2(\Omega)} \|v\|_{H_0^1(\Omega)},$$

   por lo que $F \in (H_0^1(\Omega))'$ y su norma está acotada por $\|u\|_{L^2(\Omega)}$.

En consecuencia, por el Teorema de Lax-Milgram, el problema variacional admite una única solución $y \in H_0^1(\Omega)$ para todo $u \in L^2(\Omega)$.

Así, la variable de estado t queda definida de manera implícita como una función de u, denotada $y = y(u)$. Esta dependencia será clave para derivar posteriormente las condiciones de optimalidad del problema de control.

Existencia y Unicidad del Par Óptimo

Comencemos con un resultado general que será útil más adelante, ya que podremos aplicarlo directamente a nuestro problema de control óptimo.

Supongamos que el funcional de costo $J : L^2(\Omega) \to \mathbb{R}$ se puede expresar en la forma

$$J(u) = \frac{1}{2}q(u,u) - L(u) + c,$$

donde:

- q es una forma bilineal, simétrica, continua, no negativa y coerciva sobre $L^2(\Omega)$,
- L es un funcional lineal y continuo en $L^2(\Omega)$,
- $c \in \mathbb{R}$ es una constante irrelevante para la optimización.

Bajo estas hipótesis, es posible derivar condiciones de optimalidad de primer orden que son necesarias y suficientes. Además, se puede garantizar la existencia y unicidad de una solución óptima.

Demostración: Sea $\varepsilon \neq 0$ y $v \in L^2(\Omega)$. Comparando los valores del funcional J en $\hat{u}$ y $\hat{u}+\varepsilon v$, se tiene:

$$\begin{aligned} J(\hat{u}+\varepsilon v) - J(\hat{u}) &= \left(\frac{1}{2}q(\hat{u}+\varepsilon v, \hat{u}+\varepsilon v) - L(\hat{u}+\varepsilon v) + c\right) \\ &\quad - \left(\frac{1}{2}q(\hat{u},\hat{u}) - L(\hat{u}) + c\right) \\ &= \varepsilon[q(\hat{u},v) - L(v)] + \frac{1}{2}\varepsilon^2 q(v,v), \end{aligned}$$

donde usamos la bilinealidad de q y la linealidad de L. Dividiendo por ε, obtenemos:

$$\frac{J(\hat{u}+\varepsilon v) - J(\hat{u})}{\varepsilon} = q(\hat{u},v) - L(v) + \frac{1}{2}\varepsilon q(v,v).$$

El lado izquierdo es el cociente diferencial $\frac{\Phi(\varepsilon)-\Phi(0)}{\varepsilon}$ de la función real $\Phi(\varepsilon) := J(\hat{u}+\varepsilon v)$.

Si $\hat{u}$ es un minimizador, entonces $J(\hat{u}+\varepsilon v) - J(\hat{u}) \geq 0$ para todo ε y v. Tomando el límite cuando $\varepsilon \to 0^+$, se obtiene:

$$0 \leq \Phi'(0) = \frac{d}{d\varepsilon}J(\hat{u}+\varepsilon v)\bigg|_{\varepsilon=0} = q(\hat{u},v) - L(v),$$

y tomando el límite cuando $\varepsilon \to 0^-$, se tiene:

$$0 \geq \Phi'(0) = q(\hat{u},v) - L(v).$$

Por lo tanto, debe cumplirse que

$$q(\hat{u},v) - L(v) = 0 \quad \text{para todo } v \in L^2(\Omega).$$

Recíprocamente, si $q(\hat{u},v) - L(v) = 0$ para todo $v \in L^2(\Omega)$, entonces, a partir de la identidad mostrada antes, se tiene:

$$J(\hat{u}+\varepsilon v) - J(\hat{u}) = \varepsilon[q(\hat{u},v) - L(v)] + \frac{1}{2}\varepsilon^2 q(v,v) = \frac{1}{2}\varepsilon^2 q(v,v) \geq 0 \quad \forall v \in L^2(\Omega),$$

por lo que $\hat{u}$ es un minimizador. Así, la condición de optimalidad de primer orden queda expresada como:

$$\frac{d}{d\varepsilon}J(\hat{u}+\varepsilon v)\bigg|_{\varepsilon=0} = q(\hat{u},v) - L(v) = 0 \quad \forall v \in L^2(\Omega).$$

Dado que la aplicación $v \mapsto q(\hat{u},v) - L(v)$ es un funcional lineal y continuo sobre $L^2(\Omega)$, el lado izquierdo de la ecuación define un elemento del espacio dual de $L^2(\Omega)$, el cual llamaremos la derivada de J en $\hat{u}$. Así, podemos definir la derivada de J en cualquier $u \in L^2(\Omega)$. Usaremos para ello la notación $J'(u)$, y su acción sobre una dirección v está dada por la fórmula:

$$J'(u)(v) = \frac{d}{d\varepsilon}J(u+\varepsilon v)\bigg|_{\varepsilon=0} = q(u,v) - L(v) \quad \forall v \in L^2(\Omega).$$

De esta forma, la condición de optimalidad de primer orden puede escribirse como:

$$J'(\hat{u})(v) = 0 \quad \forall v \in L^2(\Omega),$$

y recibe el nombre de ecuación de Euler.

Aún debemos verificar la existencia y unicidad de un minimizador. Para ello, observamos que la condición de optimalidad de primer orden se expresa como:

$$J'(\hat{u})(v) = q(\hat{u},v) - L(v) = 0 \quad \forall v \in L^2(\Omega).$$

Esta ecuación busca un $\hat{u} \in L^2(\Omega)$ tal que un funcional lineal se anule en todas las direcciones v. Esto puede interpretarse como un problema variacional: encontrar $\hat{u} \in L^2(\Omega)$ tal que

$$q(\hat{u},v) = L(v) \quad \forall v \in L^2(\Omega),$$

Dado que q es una forma bilineal, simétrica, continua y coerciva, y que L es un funcional lineal y continuo sobre $L^2(\Omega)$, se puede aplicar el Teorema de Lax-Milgram. Esto garantiza la existencia y unicidad de una solución $\hat{u} \in L^2(\Omega)$ al problema variacional correspondiente.

Como esta ecuación caracteriza los puntos críticos del funcional $J(u)$, y J es estrictamente convexa debido a la coercividad de q y la linealidad de L, concluimos que $\hat{u}$ es el único minimizador del problema. El estado óptimo asociado se obtiene como $\hat{y} := y(\hat{u})$, resolviendo el problema de estado con control $\hat{u}$. □

Ahora volvamos al problema de control. Recordemos que dado un control $u \in L^2(\Omega)$, el problema de estado tiene una única solución débil $y(u) \in H_0^1(\Omega)$, la cual puede ser sustituida en la expresión funcional del costo para obtener una formulación reducida que depende únicamente del control. El funcional reducido está dado por

$$J(u) := \frac{1}{2}\|y(u) - z_d\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2(\Omega)}.$$

Así, nuestro problema de minimización se reescribe como

$$\min_{u \in L^2(\Omega)} J(u).$$

Una vez determinado el control óptimo $\hat{u}$, el estado óptimo asociado se obtiene como $\hat{y} = y(\hat{u})$. Esta es la formulación en espacio reducido del problema de control óptimo.

Ahora aplicamos la estrategia anterior a nuestro problema de control óptimo, escribiendo el funcional $J(u)$ en la forma

$$J(u) = \frac{1}{2}q(u,u) - L(u) + c.$$

Para esto, desarrollemos la expresión del funcional $J(u)$:

$$\begin{aligned} J(u) &= \frac{1}{2}\|y(u) - z_d\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2(\Omega)} \\ &= \frac{1}{2}\int_\Omega (y(u)(x) - z_d(x))^2\,dx + \frac{\beta}{2}\int_\Omega u(x)^2\,dx \\ &= \frac{1}{2}\int_\Omega y(u)(x)^2\,dx - \int_\Omega y(u)(x)z_d(x)\,dx + \frac{1}{2}\int_\Omega z_d(x)^2\,dx + \frac{\beta}{2}\int_\Omega u(x)^2\,dx. \end{aligned}$$

Definiendo entonces:

$$\begin{aligned} q(u,v) &:= \int_\Omega y(u)(x)y(v)(x)\,dx + \beta\int_\Omega u(x)v(x)\,dx, \\ L(v) &:= \int_\Omega z_d(x)y(v)(x)\,dx, \\ c &:= \frac{1}{2}\int_\Omega z_d(x)^2\,dx, \end{aligned}$$

se obtiene que

$$J(u) = \frac{1}{2}q(u,u) - L(u) + c,$$

como queríamos. A continuación, verificaremos que las hipótesis necesarias para aplicar los resultados generales se cumplen en nuestro caso:

- **q es una forma bilineal, simétrica, continua, no negativa y coerciva sobre $L^2(\Omega)$:**

  Sea $y(u)$ la solución débil del problema de estado asociado al control $u \in L^2(\Omega)$. Como la ecuación de estado es lineal y bien planteada, el operador que asocia $u \mapsto y(u)$ es lineal y continuo de $L^2(\Omega)$ a $H^1_0(\Omega)$, e incluso a $L^2(\Omega)$.

  La bilinealidad y simetría de q se siguen directamente de la definición:

  $$q(u,v) = \int_\Omega y(u)y(v)\,dx + \beta\int_\Omega uv\,dx,$$

  pues ambos términos son bilineales y simétricos.

  Para la continuidad, usamos las desigualdades de Cauchy-Schwarz y la continuidad del operador $u \mapsto y(u)$:

  $$|q(u,v)| \le \|y(u)\|_{L^2(\Omega)}\|y(v)\|_{L^2(\Omega)} + \beta\|u\|_{L^2(\Omega)}\|v\|_{L^2(\Omega)} \le C\|u\|_{L^2(\Omega)}\|v\|_{L^2(\Omega)},$$

  para cierta constante $C > 0$.

  La no negatividad se deduce porque

  $$q(u,u) = \int_\Omega y(u)^2\,dx + \beta\int_\Omega u^2\,dx \ge 0.$$

  La coercividad se obtiene directamente del segundo término:

  $$q(u,u) \ge \beta\|u\|^2_{L^2(\Omega)},$$

  ya que $\beta > 0$. Por lo tanto, q es coerciva en $L^2(\Omega)$.

- **L es un funcional lineal y continuo en $L^2(\Omega)$:**

  Por definición,

  $$L(v) = \int_\Omega z_d\, y(v)\, dx.$$

  La linealidad de L se sigue de la linealidad de $v \mapsto y(v)$. Para la continuidad, usamos Cauchy-Schwarz y la continuidad de $y(v)$:

  $$|L(v)| \leq \|z_d\|_{L^2(\Omega)} \|y(v)\|_{L^2(\Omega)} \leq C\|v\|_{L^2(\Omega)},$$

  para alguna constante $C > 0$.

- **$c \in \mathbb{R}$ es una constante:**

  Se tiene que

  $$c = \frac{1}{2}\int_\Omega z_d^2\, dx,$$

  lo cual es una constante escalar.

Entonces, se concluye la existencia y unicidad de un control óptimo $\hat{u} \in L^2(\Omega)$, caracterizado por la ecuación de Euler (con $\hat{y} = y(\hat{u})$):

$$J'(\hat{u})v = q(\hat{u}, v) - L(v) = \int_\Omega (\hat{y} - z_d)y(v)\, dx + \beta \int_\Omega \hat{u}v = 0 \quad \forall v \in L^2(\Omega).$$

Estado Adjunto

El estado adjunto se utiliza para representar el gradiente del funcional de costo $\nabla J(u)$ para este problema. Si bien la ecuación de Euler caracteriza al control óptimo $\hat{u}$, resolver este sistema con métodos computacionales es costoso. Esto se debe a que para calcular la derivada direccional del funcional J, es necesario evaluar una función de la forma $y(v)$, lo que involucra resolver la ecuación de estado para cada dirección v. Con esta ecuación no es posible llegar a una expresión para calcular directamente el gradiente $\nabla J(\hat{u})$. Para resolver este problema se introduce una reformulación que introduce un multiplicador $p \in H_0^1(\Omega)$, también llamado *estado adjunto*, que permite reescribir el funcional de costo de forma manipulable computacionalmente. Se plantea un funcional aumentado de la forma:

$$J(u) = \frac{1}{2}\int_\Omega (y(u) - z_d)^2\, dx + \frac{\beta}{2}\int_\Omega u^2\, dx + \int_\Omega (f + u)p - \nabla y(u) \cdot \nabla p dx,$$

donde el término adicional es nulo para todo u debido a que esta es una formulación débil de la restricción de u. Utilizando la descomposición $y(u) = y_0(u) + y(0)$, donde $y_0(u)$ es la "parte lineal" de $y(u)$, se simplifica el funcional lineal asociado a:

$$\tilde{L}u = \int_\Omega y_0(u)(z_d - y(0))\, dx + \int_\Omega up - \nabla y_0(u) \cdot \nabla p dx.$$

A partir de aquí, se obtiene una expresión para la derivada direccional:

$$J'(\bar{u})v = \int_\Omega (y(\bar{u}) - z_d)y_0(v)\, dx + \int_\Omega (p + \beta\bar{u})v\, dx - \int_\Omega \nabla y_0(v) \cdot \nabla p dx,$$

la cual aún depende de $y_0(v)$, por lo que no es computacionalmente factible debido a la necesidad de reevaluar para distintos v. Para resolver esta desventaja se define el problema adjunto, que consiste en buscar $p \in H_0^1(\Omega)$ tal que

$$\int_\Omega \nabla\psi \cdot \nabla p dx = \int_\Omega (y(\bar{u}) - z_d)\psi \, dx \quad \text{para todo } \psi \in H_0^1(\Omega).$$

Esta es la formulación débil del sistema elíptico:

$$\begin{cases} -\Delta p = y(\bar{u}) - z_d & \text{en } \Omega, \\ p = 0 & \text{en } \Gamma. \end{cases}$$

Dado que $y(\bar{u}) - z_d \in L^2(\Omega)$, se puede utilizar el teorema de Lax-Milgram para mostrar que este problema tiene solución única $\bar{p}$. Al sustituir esta solución en la expresión de $J'(\bar{u})v$, los términos asociados a $y_0(v)$ se anulan, obteniendo:

$$J'(\bar{u})v = \int_\Omega (\bar{p} + \beta\bar{u})v \, dx,$$

lo que implica que el *gradiente del funcional* está dado por:

$$\nabla J(\bar{u}) = \bar{p} + \beta\bar{u}.$$

Este resultado se utiliza para resolver el problema con métodos de descenso. El estado adjunto $\bar{p}$ es útil para poder computar el gradiente funcional sin resolver múltiples veces la ecuación de estado. En el caso particular donde $\bar{u} = \hat{u}$, es decir, si estamos evaluando en el control óptimo, la ecuación de Euler se reduce a:

$$\int_\Omega (\hat{p} + \beta\hat{u})v \, dx = 0 \quad \text{para todo } v \in L^2(\Omega),$$

lo que implica directamente:

$$\hat{p} + \beta\hat{u} = 0 \quad \text{o equivalentemente} \quad \hat{u} = -\frac{1}{\beta}\hat{p} \quad \text{casi en todas partes en } \Omega.$$

Lo anterior se puede expresar en la siguiente formulación de condiciones de optimalidad para el problema de control. El par $(\hat{u}, \hat{y})$ es óptimo si y sólo si existen funciones $\hat{y}$ y $\hat{p}$ que satisfacen simultáneamente:

$$\begin{cases} -\Delta\hat{y} = \hat{u} & \text{en } \Omega, \\ -\Delta\hat{p} = \hat{y} - z_d & \text{en } \Omega, \\ \hat{p} + \beta\hat{u} = 0 & \text{en } \Omega, \\ \hat{y} = \hat{p} = 0 & \text{en } \Gamma. \end{cases} \tag{1}$$

De esta forma se puede caracterizar el problema de control óptimo de una forma eficiente computacionalmente.

Discretización del Problema

Existen múltiples alternativas para implementar computacionalmente un algoritmo que resuelva aproximadamente este problema, cada uno con ventajas y desventajas. Es posible caracterizar las condiciones de la solución óptima del problema como un sistema de EDPs, que puede ser discretizado posteriormente, así como discretizar el dominio de las variables del problema original de optimización y resolver la formulación resultante. Estas estrategias se conocen como *optimize-then-discretize* (OtD) y *discretize-then-optimize* (DtO), y en principio la solución obtenida con ambos métodos no es necesariamente idéntica. Se analiza la construcción de estos métodos y se estudia la conmutatividad de las fases de optimización y discretización.

Optimizar y Luego Discretizar

Escribimos el sistema que incluye el estado adjunto en formulación débil (1) como fue mostrado anteriormente

$$\begin{cases} a(y,\varphi) = (u,\varphi)_{L^2(\Omega)} & \forall \varphi \in V \\ a^*(p,\psi) = -(y - z_d, \psi)_{L^2(\Omega)} & \forall \psi \in V \\ (\beta u - p, v)_{L^2(\Omega)} = 0 & \forall v \in \mathcal{U}. \end{cases} \tag{2}$$

Esto para $y \in V$, $u \in \mathcal{U}$ y $p \in V$ y las formas bilineales a, a^* corresponden al producto interno de los gradientes de las funciones, lo que representa la acción del laplaciano en la formulación débil. Se reparametrizó p con el signo contrario para que la discretización numérica tenga una estructura más conveniente. Elegimos espacios de dimensión finita V_h y $\mathcal{U}_h$ en base a triangulaciones del dominio de las variables, que asumiremos que se pueden llevar a cabo con la elección del dominio Ω realizada. Obtenemos el sistema

$$\begin{cases} a(y_h,\varphi_h) = (u_h,\varphi_h)_{L^2(\Omega)} & \forall \varphi_h \in V \\ a^*(p_h,\psi_h) = -(y_h - z_d, \psi_h)_{L^2(\Omega)} & \forall \psi \in V_h \\ (\beta u_h - p_h, v_h)_{L^2(\Omega)} = 0 & \forall v_h \in \mathcal{U}_h \end{cases} \tag{3}$$

Donde y_h, p_h y u_h son combinaciones lineales de las bases $\{\varphi_j\}_{j=1}^{N_y} \subseteq V_h$ y $\{\psi_j\}_{j=1}^{N_u} \subseteq \mathcal{U}_h$. Al expresar (3) como un sistema matricial obtenemos lo siguiente

$$\begin{bmatrix} \mathbb{M} & 0 & \mathbb{A}^\intercal \\ 0 & \beta\mathbb{N} & -\mathbb{B}^\intercal \\ \mathbb{A} & -\mathbb{B} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbb{M}\mathbf{z}_d \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{4}$$

donde los elementos de cada matriz están dados por las siguientes expresiones:

$$(\mathbb{A})_{ij} = a(\varphi_j, \varphi_i), \quad (\mathbb{B})_{ij} = (\psi_j, \varphi_i)_{L^2(\Omega)}, \quad (\mathbb{M})_{ij} = (\varphi_j, \varphi_i)_{L^2(\Omega)}, \quad (\mathbb{N})_{ij} = (\psi_j, \psi_i)_{L^2(\Omega)}.$$

Notamos que cuando se eligen espacios y elementos de la base equivalentes para V_h y $\mathcal{U}_h$ obtenemos que $\mathbb{B} = \mathbb{M} = \mathbb{N}$.

Discretizar y Luego Optimizar

Esta estrategia se basa en sustituir los espacios de las variables del problema de control original por sus discretizaciones en dimensión finita y resolver el problema de optimización en este dominio, el problema resultante es

$$\begin{aligned} \text{mín } \tilde{J}_h(y_h, u_h) :=& \frac{1}{2}\int_\Omega (y_h - z_d)^2 dx + \frac{\beta}{2}\int_\Omega u_h^2 dx \\ \text{s.a.} \quad & u_h \in \mathcal{U}_h \\ y_h \in V_h : \quad & a(y_h, \varphi_h) = (u_h, \varphi_h)_{L^2(\Omega)} \quad \forall \varphi_h \in V_h. \end{aligned}$$

Lo anterior puede ser escrito en forma vectorial considerando la notación para las matrices presentada anteriormente

$$\begin{aligned} \text{mín } \tilde{J}_h(\mathbf{y},\mathbf{u}) &= \frac{1}{2}(\mathbf{y}-\mathbf{z}_d)^\intercal \mathbb{M}(\mathbf{y}-\mathbf{z}_d) + \frac{\beta}{2}\mathbf{u}^\intercal \mathbb{N}\mathbf{u} \\ \text{s.a.} \quad & \mathbf{u} \in \mathbb{R}^{N_u} \\ \mathbf{y} \in \mathbb{R}^{N_y} : \quad & \mathbb{A}\mathbf{y} = \mathbb{B}\mathbf{u}. \end{aligned}$$

Este problema de optimización puede ser resuelto de forma estable utilizando métodos iterativos usuales de optimizacíon convexa como *steepest descent*, aunque se puede encontrar la solución óptima utilizando condiciones de primer orden sobre los costos reducidos del problema de optimización, que corresponden a la siguiente expresión

$$J_h(\mathbf{u}) = \tilde{J}_h(\mathbf{y}(\mathbf{u}),\mathbf{u}) = \tilde{J}_h(\mathbb{A}^{-1}\mathbb{B}\mathbf{u},\mathbf{u})$$

Las condiciones de optimalidad corresponden al siguiente sistema lineal

$$\begin{cases} \mathbb{A}\mathbf{y} = \mathbb{B}\mathbf{u} \\ \mathbb{A}^\intercal \mathbf{p} = -\mathbb{M}(\mathbf{y}-\mathbf{z}_d) \\ \beta\mathbb{N}\mathbf{u} - \mathbb{B}^\intercal \mathbf{p} = \mathbf{0} \end{cases}. \tag{5}$$

Este sistema es equivalente a la estrategia OtD debido a la elección de espacios discretizados, en la siguiente sección se muestra que esto no es así.

Conmutatividad de las Estrategias

Generalizamos la formulación OtD para elegir espacios distintos entre las variables $\mathbf{y}$ y $\mathbf{p}$ para mostrar un caso en que no existe conmutatividad entre las estrategias. Conservamos los espacios $u \in \mathcal{U}_h$ y $y \in V_h$, e introducimos un nuevo espacio discreto $\tilde{V}_h$ de dimensión N_p para $\mathbf{p}$ con base $\{\tilde{\varphi}_j\}_{j=1}^{N_p}$, lo que define las siguientes matrices

$$(\tilde{\mathbb{A}})_{ij} = a(\tilde{\varphi}_j, \tilde{\varphi}_i), \quad (\tilde{\mathbb{T}})_{ij} = (\varphi_j, \tilde{\varphi}_i)_{L^2(\Omega)}, \quad (\tilde{\mathbb{B}})_{ij} = (\psi_j, \tilde{\varphi}_i)_{L^2(\Omega)}$$

El sistema resultante corresponde a

$$\begin{bmatrix} \tilde{\mathbb{T}} & 0 & \tilde{\mathbb{A}} \\ 0 & \beta\mathbb{N} & -\tilde{\mathbb{B}}^\intercal \\ \mathbb{A} & -\mathbb{B} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbb{T}}\mathbf{z}_d \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{6}$$

En este caso se pierde la simetría ya que las matrices $\tilde{\mathbb{A}}$ y $\mathbb{A}$ provienen de espacios distintos. Además de perder regularidad, este sistema entrega soluciones distintas al de la estrategia DtO ya que en (5) existe simetría al igual que al elegir $\tilde{V}_h = V_h$. Se concluye que estas metodologías no conmutan en general.

Convergencia de las Discretizaciones

Se muestra convergencia al recurrir a los resultados disponibles para problemas de control óptimo escalares con una restricción basada en una ecuación elíptica, esto se logra expresando el sistema de la solución óptima como un problema de punto silla. Definimos $\mathbf{x} = (y,u) \in X$ donde $X = V \times \mathcal{U}$. Entonces

$$(\mathbf{x},\mathbf{w})_X = (y,z)_V + (u,v)_{\mathcal{U}} \quad \forall \mathbf{x} = (y,u), \mathbf{w} = (z,v) \in X$$

Definimos las formas bilineales para el problema agregado en el espacio producto, dadas por

$$\mathcal{D}(\mathbf{x},\mathbf{w}) = (y,z)_{L^2(\Omega)} + \beta(u,v)_{L^2(\Omega)}$$
$$\mathcal{E}(q,\mathbf{x}) = a(y,q) - (u,q)_{L^2(\Omega)}$$

Donde $\beta \geq 0$ es el parámetro regularizador de la norma del control. A partir del teorema demostrado para problemas de punto silla en tareas del curso se concluye que este problema está bien puesto. Además, el Teorema 6.3 de [1] nos dice lo siguiente

Teorema 6.3. *La siguiente estimación a posteriori ponderada del error se cumple para el funcional de costo* $\tilde{J}$,

$$\left|\tilde{J}(\hat{y},\hat{u}) - \tilde{J}(\hat{y}_h,\hat{u}_h)\right| \leq \eta_{\omega_\rho}(\hat{y}_h,\hat{u}_h,\hat{p}_h) = \sum_{K\in\mathcal{T}_h} \eta_K(\hat{y}_h,\hat{u}_h,\hat{p}_h)$$

donde

$$\eta_K(\hat{y}_h,\hat{u}_h,\hat{p}_h) = \omega_K^p\rho_K^y + \omega_K^y\rho_K^p + \omega_K^u\rho_K^u$$

es el indicador de error local, con

$$\rho_K^y = \|R_y(\hat{y}_h)\|_K + h_K^{-1/2}\|r_y(\hat{y}_h)\|_{\partial K},$$

$$\rho_K^p = \|R_p(\hat{p}_h)\|_K + h_K^{-1/2}\|r_p(\hat{p}_h)\|_{\partial K},$$

$$\rho_K^u = \|R_u(\hat{y}_h)\|_K.$$

En este caso las funciones $R_y(\hat{y}_h), R_p(\hat{p}_h), r_y(\hat{y}_h), r_p(\hat{p}_h), R_u(\hat{p}_h,\hat{u}_h)$ están definidas de la siguiente forma

$$R_y(\hat{y}_h) = \hat{u}_h + \Delta\hat{y}_h, \quad r_y(\hat{y}_h) = -\frac{1}{2}\left[\frac{\partial\hat{y}_h}{\partial n}\right]_{\Delta K\setminus\Gamma},$$
$$R_p(\hat{p}_h) = \hat{y}_h - z_d + \Delta\hat{p}_h, \quad r_p(\hat{p}_h) = -\frac{1}{2}\left[\frac{\partial\hat{p}_h}{\partial n}\right]_{\Delta K\setminus\Gamma},$$
$$R_u(\hat{p}_h,\hat{u}_h) = \hat{p}_h,$$

Y los pesos $\omega_K^p, \omega_K^y, \omega_K^u$ se definen como

$$\omega_K^p = \|\hat{p}-\hat{p}_h\|_K + h_K^{1/2}\|\hat{p}-\hat{p}_h\|_{\partial K},$$
$$\omega_K^y = \|\hat{y}-\hat{y}_h\|_K + h_K^{1/2}\|\hat{y}-\hat{y}_h\|_{\partial K},$$
$$\omega_K^p = \|\hat{u}-\hat{u}_h\|_K,$$

Si resolvemos el problema para $V = \mathcal{U} = H^1(\Omega)$, entonces podemos caracterizar los errores de interpolación de una malla de elementos finitos en $\mathbb{P}_1$ con el resultado visto en el curso. En principio tendremos que

$$\|\hat{p}-\hat{p}_h\|_{H^0(\Omega)}, \|\hat{y}-\hat{y}_h\|_{H^0(\Omega)}, \|\hat{u}-\hat{u}_h\|_{H^0(\Omega)} \in \mathcal{O}(h)$$

Luego, al considerar a los funcionales R, r como acotados tenemos que

$$\eta_K(\hat{y}_h,\hat{u}_h,\hat{p}_h) \in \mathcal{O}(h^{1/2})$$

Variable de Control Restringida

Es completamente razonable desde el punto de vista físico restringir la variable de control u. En muchos problemas de control óptimo, es importante tener en cuenta las limitaciones naturales de las variables de control, como las capacidades de los actuadores o los límites de energía. Así, la introducción de una restricción como $-a \leq u \leq a$ refleja una limitación física fundamental, asegurando que el control no exceda los límites permitidos por el sistema.

El nuevo modelo sugerido para este problema de control óptimo es el siguiente:

$$\begin{aligned} \min_{y,u} \quad & \frac{1}{2}\|y - z_d\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2(\Omega)} \\ \text{s.a.} \quad & -\Delta y = u \quad \text{en } \Omega, \\ & u = 0 \quad \text{en } \partial\Omega, \\ & -a \leq u \leq a \quad \text{en } \Omega, \end{aligned}$$

donde a es un parámetro que define los límites máximos y mínimos del control.

Estudiemos este nuevo problema. Utilizando herramientas de convexidad es posible mostrar lo siguiente partiendo de la condición suficiente para optimalidad del problema en dimensión infinita

$$\hat{u} \in \mathcal{U}_{ad} \text{ es un control óptimo } \iff J'(\hat{u})(v - \hat{u}) = (y(\hat{u}) - z_d, y(v) - y(\hat{u}))_Z + \beta\,(\hat{u}, v - \hat{u})_U \geq 0 \quad \forall v \in \mathcal{U}_{ad}.$$

Queremos demostrar existencia y unicidad del control óptimo en este contexto. El teorema de Stampacchia es una herramienta útil para este caso. Sea $\mathcal{U}$ un espacio de Hilbert, $K \subseteq \mathcal{U}$ un conjunto cerrado y convexo, y q una forma bilineal simétrica, continua y coerciva sobre $\mathcal{U}$. Es decir, existen números positivos N, θ tales que:

$$|q(u, v)| \leq N\|u\|_{\mathcal{U}}\|v\|_{\mathcal{U}} \quad \forall u, v \in \mathcal{U}$$

y

$$q(u, u) \geq \theta\|u\|^2_{\mathcal{U}} \quad \forall u \in K.$$

Entonces, si $L \in \mathcal{U}^*$, la desigualdad variacional

$$q(u, v - u) \geq L(v - u) \quad \forall v \in K$$

tiene una única solución $u \in K$. En este caso, $q(\hat{u}, v - \hat{u}) = (y(\hat{u}) - z_d, y(v) - y(\hat{u}))_Z + \beta(\hat{u}, v - \hat{u})_U$, por lo que se cumplen las condiciones de elipticidad y continuidad del operador solo si el dominio de controles admisibles es acotado. Podemos introducir una restricción del siguiente tipo para utilizar este teorema

$$u \in [-U, U]$$

Generando de esta forma un dominio cerrado, acotado y convexo. Se concluye entonces que existe una solución óptima única para el problema de control óptimo restringido.

Método del Gradiente

En el método DtO podemos utilizar la igualdad $\mathbb{A}\mathbf{y} = \mathbb{B}\mathbf{u}$ para parametrizar la variable $\mathbf{y}$ en términos de u imponiendo la restricción que existe entre estas variables. Luego, para optimizar $\mathbf{u}$ consideramos

la función objetivo como $J_h(\mathbf{u}) := J_h(\mathbb{A}^{-1}\mathbb{B}\mathbf{u}, \mathbf{u})$ (función de costos reducidos) y utilizamos su gradiente respecto a $\mathbf{u}$ para encontrar la dirección de descenso. Este gradiente está dado por

$$\nabla J_h(\mathbf{u}) = \beta\mathbb{N}\mathbf{u} + \mathbb{B}^\intercal(\mathbb{A}^{-\intercal}\mathbb{M}(\mathbb{A}^{-1}\mathbb{B}\mathbf{u} - \mathbf{z}_d)) = \beta\mathbb{N}\mathbf{u} - \mathbb{B}^\intercal\mathbf{p}$$

Si nos movemos en la dirección de descenso, actualizando la variable de control $\mathbf{u}$ en la dirección opuesta al gradiente de la función objetivo, podemos esperar que el proceso de optimización se acerque al óptimo bajo ciertas condiciones. Dado que el problema es estrictamente convexo, sabemos que la función objetivo $J_h(\mathbf{u})$ tiene un único mínimo global, y cualquier trayectoria de descenso, si se elige adecuadamente el tamaño del paso, llevará eventualmente a esta solución óptima.

Sin embargo, la convergencia efectiva depende de la elección de los pasos en cada iteración. Si el tamaño del paso es demasiado grande, es posible que el algoritmo no converja o que incluso se aleje del óptimo, mientras que un paso demasiado pequeño podría hacer que el algoritmo se quede atrapado en un proceso de optimización muy lento, sin llegar al óptimo en un tiempo razonable. Por lo tanto, la selección adecuada de los pasos es crucial para garantizar que el algoritmo se mueva hacia el mínimo global.

El método de búsqueda lineal tipo Armijo, que ajusta el tamaño del paso de acuerdo con la disminución de la función objetivo, es una estrategia eficaz para asegurarse de que el algoritmo siga una dirección de descenso adecuada en cada iteración. En la práctica, si se eligen pasos suficientemente pequeños cuando el gradiente es grande, y si el tamaño del paso se ajusta de manera eficiente en cada iteración, el algoritmo convergerá al mínimo global. En resumen, la convergencia al óptimo se garantiza si se elige un tamaño de paso apropiado en cada iteración, lo que es fundamental en problemas estrictamente convexos.

El método del gradiente también puede ser extendido para problemas de optimización con restricciones. En este caso, dado que el método de gradiente es adecuado para problemas sin restricciones, el desafío radica en asegurar que las iteraciones del algoritmo permanezcan dentro del conjunto factible de soluciones, es decir, dentro del dominio definido por las restricciones.

Para abordar este problema, podemos aplicar la proyección sobre el conjunto factible en cada iteración. Esto significa que, después de actualizar la variable de control $\mathbf{u}$ siguiendo la dirección de descenso, proyectamos el resultado sobre el conjunto de restricciones para asegurarnos de que la nueva iteración también sea factible. La operación de proyección garantiza que la variable de control $\mathbf{u}$ esté dentro del dominio restringido, sin necesidad de modificar la dirección de descenso.

Matemáticamente, si el conjunto de restricciones es C, después de calcular el nuevo valor $\mathbf{u}^{k+1}$ mediante el método de gradiente, podemos obtener el nuevo valor factible $\mathbf{u}^{k+1}_{\text{proj}} = \text{Proj}_C(\mathbf{u}^{k+1})$, donde Proj_C denota la proyección sobre el conjunto C. Esta proyección ajusta el valor de $\mathbf{u}^{k+1}$ para que se cumplan las restricciones.

Lo importante aquí es que, al usar la proyección, mantenemos las mismas garantías teóricas del método del gradiente sin restricciones. Es decir, la estricta convexidad del problema y la correcta elección del paso aseguran la convergencia al mínimo global del problema restringido, siempre y cuando el paso en cada iteración sea seleccionado adecuadamente. La proyección no altera la naturaleza del problema, ya que, si se proyecta sobre un conjunto convexo y cerrado, el conjunto sigue siendo un dominio convexo, lo que preserva las propiedades de suavidad y convexidad necesarias para la convergencia.

En nuestro caso, debemos proyectar la variable de control $\mathbf{u}$ para que permanezca dentro del dominio factible, esto es, el intervalo restringido $-a \leq \mathbf{u} \leq a$. Este conjunto es un conjunto convexo y cerrado, lo que facilita la proyección de la siguiente manera:

Dado un vector $\mathbf{u} = (u_1, u_2, \ldots, u_n)$, la proyección sobre el conjunto $C = \{\mathbf{u} \in \mathbb{R}^n : -a \leq u_i \leq a, \forall i\}$ consiste en ajustar cada componente de $\mathbf{u}$ para que cumpla con la restricción de $-a \leq u_i \leq a$ de manera independiente. Esto se puede calcular de forma sencilla para cada componente u_i de $\mathbf{u}$ mediante la siguiente operación:

$$\text{Proj}_C(u_i) = \begin{cases} -a & \text{si } u_i < -a, \\ u_i & \text{si } -a \leq u_i \leq a, \\ a & \text{si } u_i > a. \end{cases}$$

Por lo tanto, la proyección de $\mathbf{u}$ sobre el conjunto C es simplemente la aplicación de esta operación a cada componente u_i de $\mathbf{u}$. Es decir, la proyección de $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ sobre el conjunto restringido $-a \leq u_i \leq a$ se puede escribir como:

$$\text{Proj}_C(\mathbf{u}) = (\text{Proj}_C(u_1), \text{Proj}_C(u_2), \ldots, \text{Proj}_C(u_n)),$$

donde la proyección de cada u_i es calculada independientemente.

En resumen, al proyectar después de cada actualización del gradiente, el método sigue siendo válido y mantiene las mismas garantías teóricas que en el caso sin restricciones. De esta manera, podemos implementar una versión restringida del método del gradiente, asegurando que la solución converja de manera eficiente al mínimo global del problema restringido.

Simulaciones

Comencemos con las simulaciones computacionales.

Caso Irrestricto

A continuación, pasaremos a realizar las simulaciones computacionales que permiten validar los resultados teóricos. Para ello, implementaremos el descenso de gradiente con búsqueda lineal de Armijo, utilizando Python como lenguaje de programación.

Comenzamos por importar las librerías necesarias para la implementación. A continuación, se muestra el bloque de código correspondiente:

```
#Operaciones matriciales.
import numpy as np
#Crear el espacio de aproximacion.
from skfem import *
from skfem.models.poisson import laplace, mass
#Resolver los sistemas lineales. Lo hace a trav s de factorizaciones LU
from scipy.sparse.linalg import spsolve
#Graficar.
import matplotlib.pyplot as plt
from matplotlib.ticker import ScalarFormatter
```

A continuación, definiremos la función objetivo $z_d(x, y)$, la cual representa el estado deseado al que se pretende aproximar la solución del sistema. Esta función cumple con condiciones de tipo Dirichlet homogéneas, ya que se anula en la frontera del dominio $[0, 1] \times [0, 1]$, y además es suave en el interior del dominio. En el siguiente bloque se presenta su implementación, así como la evaluación de esta función sobre los grados de libertad del espacio base.

```
#Definimos la funcion objetivo z_d(x, y)
def z_d(x, y):
    return 10 * x * (1 - x) * y * (1 - y)

#Evaluamos z_d en las coordenadas de los grados de libertad de la base.
```

```python
#Se separan las coordenadas x e y, y luego se evalua z_d en todos los dofs.
def Getzd(basis):
    x = basis.doflocs[0, :]  #Coordenadas x.
    y = basis.doflocs[1, :]  #Coordenadas y.
    return z_d(x, y)         #Evaluacion de z_d en (x, y).
```

En lo que sigue, implementamos las funciones y ecuaciones obtenidas en el desarrollo del procedimiento de optimización. Estas incluyen la resolución de la ecuación de estado para obtener el estado $y(u)$, la resolución de la ecuación adjunta para obtener el multiplicador $p(y)$, la evaluación funcional del costo $J(u, y(u))$, y el cálculo del gradiente de esta funcional con respecto al control. Todas estas funciones serán fundamentales para llevar a cabo el método de descenso del gradiente en la siguiente sección.

```python
#y(u) a traves de la ecuacion de estado.
def gety(A, M, u):
    y = spsolve(A, M @ u)
    return y

#p(y) a traves de la ecuacion adjunta.
def getp(A, M, zd, y):
    return spsolve(A, -M @ (y - zd))

#Funcion objetivo J(u, y(u)).
def J(M, beta, zd, y, u):
    return 0.5 * (y - zd).T @ M @ (y - zd) + 0.5 * beta * u.T @ M @ u

#Gradiente de la funcion objetivo
def grad_J(M, beta, u, p):
    return beta * M @ u - M @ p
```

Para llevar a cabo la simulación, es necesario primero construir el espacio de elementos finitos sobre el cual resolveremos el problema. Utilizaremos una malla triangular uniforme en el dominio $[0, 1]^2$, que refinaremos un número determinado de veces.

El espacio de aproximación estará conformado por funciones lineales por partes (elementos $\mathbb{P}_1$), lo que nos permitirá ensamblar las matrices de rigidez (asociada al operador Laplaciano y de masa (asociada al producto escalar en L^2). Además, impondremos condiciones de Dirichlet homogéneas en el borde del dominio identificando los nodos que se encuentran sobre los bordes del cuadrado y modificando las filas y columnas correspondientes en la matriz de rigidez.

En el caso particular con nivel de refinamiento `ref` $= 6$, se generan exactamente 16641 nodos y 32768 elementos triangulares, lo que garantiza una discretización suficientemente fina para capturar la solución con buena precisión.

```python
#Generamos el espacio de aproximacion
def getSpace(ref):
    #Generamos una malla triangular en el dominio unitario, refinada ref veces.
    mesh = MeshTri().refined(ref)

    #Definimos el espacio de aproximacion usando funciones lineales por partes.
    basis = Basis(mesh, ElementTriP1())

    #Ensamblamos la matriz de rigidez.
    A = asm(laplace, basis).tocsc()

```

```
    #Ensamblamos la matriz de masa.
    M = asm(mass, basis).tocsc()

    #Obtenemos los grados de libertad.
    N = basis.N

    print(f"Hay {mesh.p.shape[1]} nodos y {mesh.t.shape[1]} elementos")

    #Aqui guardaremos el ID de los nodos del borde.
    dofs = []

    #Obtenemos las coordenadas de los nodos.
    nodes = basis.doflocs.T

    #Recorremos cada nodo y verificamos si esta en el borde.
    for i, (x, y) in enumerate(nodes):
        if (abs(x) < 1e-10 or abs(x-1) < 1e-10 or abs(y) < 1e-10 or abs(y-1) < 1
            e-10):
             dofs.append(i)

    #Lo convertimos a un arreglo de NumPy.
    dofs = np.array(dofs, dtype=int)

    #Dejamos A en formato LIL para poder modificarla.
    A = A.tolil()

    #Imponemos condiciones de Dirichlet homogeneas en el borde.
    for dof in dofs:
        A[:, dof] = 0
        A[dof, dof] = 1

    #Reconvertimos A a formato CSC.
    A = A.tocsc()

    #Retornamos los objetos necesarios
    return basis, A, M, N
```

A continuación, implementamos el algoritmo de descenso de gradiente con búsqueda lineal de Armijo para resolver el problema de control óptimo.

Los parámetros utilizados en la implementación son: $\sigma = 10^{-2}$ para la condición de Armijo, $\rho = 0{,}5$ como factor de reducción del paso, $t_0 = 1{,}0$ como paso inicial, $t_{\text{mín}} = 10^{-20}$ como paso mínimo permitido, $\text{tol} = 10^{-18}$ como tolerancia relativa de parada basada en la norma del gradiente, $K_{\text{máx}} = 200$ como número máximo de iteraciones, y un nivel de refinamiento $ref = 7$, lo que genera una malla suficientemente fina para una aproximación precisa.

```
#Implementamos el metodo del gradiente
def GradientMethod(beta, sigma=1e-2, rho=0.5, t0=1.0, tmin=1e-20, tol=1e-18,
    Kmax=200, ref = 7):
    #Obtenemos los elementos necesarios del espacio de aproximacion.
    basis, A, M, N = getSpace(ref)

    #Obtenemos la proyeccion de la funcion objetivo sobre el espacio aproximado.
    zd = Getzd(basis)
```

```
8
9       #Inicializamos el control u de forma aleatoria.
10      u = np.random.rand(N)
11      #Obtenemos y(u) de la ecuacion de estado.
12      y = gety(A, M, u)
13      #Obtenemos p(y) de la ecuacion adjunta.
14      p = getp(A, M, zd, y)
15      #Obtenemos el gradiente de la funcion objetivo.
16      grad = grad_J(M, beta, u, p)
17      #Obtenemos la norma inicial del gradiente para usarla de referencia.
18      norm_grad0 = np.linalg.norm(grad)
19      #Obtenemos el valor de la funcion objetivo en el punto inicial.
20      J_val = J(M, beta, zd, y, u)
21
22      #Inicializamos el contador de iteraciones.
23      k = 0
24      #Inicializamos el paso del metodo de descenso de gradiente.
25      t = t0
26      while np.linalg.norm(grad) / norm_grad0 > tol and k < Kmax:
27          #Como no reinicializamos el paso, probamos amplificarlo una vez.
28          t /= rho
29
30          #Intentamos dar un paso usando el valor t actual.
31          while t > tmin:
32              #Actualizamos de u usando el gradiente y el paso t.
33              u_new = u - t * grad
34              #Obtenemos y(u) de la ecuacion de estado.
35              y_new = gety(A, M, u_new)
36              #Obtenemos p(y) de la ecacion adjunta.
37              p_new = getp(A, M, zd, y_new)
38              #Calculamos el nuevo valor de la funcion objetivo.
39              J_new = J(M, beta, zd, y_new, u_new)
40
41              #Verificamos la condicion de Armijo.
42              if J_new <= J_val - sigma * t * (grad @ grad):
43                  #Actualizamos u, y, p, J y su gradiente con los nuevos valores.
44                  u = u_new
45                  y = y_new
46                  p = p_new
47                  grad = grad_J(M, beta, u, p)
48                  J_val = J_new
49                  #Se incrementa el contador de iteraciones.
50                  k += 1
51                  #Obtenemos informacion relevante para validar que esta
                        funcionando.
52                  print(f"Iter {k}: J = {J_val:.6e}, ||grad|| = {np.linalg.norm(
                        grad):.2e}")
53                  break
54              else:
55                  #Si la condicion de Armijo no se cumple, se reduce el paso t y
                        se vuelve a intentar.
56                  t *= rho
57          else:
```

```
            #Si no se encuentra un valor adecuado para t, se termina.
            print("Backtracking fallo.")
            break

    #Se devuelve el espacio de aproximacion, junto con la solucion
    return basis, zd, u, y, p
```

Finalmente, implementamos una función auxiliar para visualizar gráficamente las soluciones obtenidas: la función objetivo proyectada z_d, el control óptimo u, el estado y y el adjunto p. Esta visualización permite interpretar y comparar espacialmente las distintas funciones involucradas en el problema de control óptimo.

```
#Creamos una funcion para graficar la solucion
def show(basis, zd, u, y, p):

    #Obtenemos las coordenadas de la base
    x = basis.doflocs[0, :]
    y_coords = basis.doflocs[1, :]

    #Crear una figura con 4 subplots
    fig, axs = plt.subplots(2, 2, figsize=(16, 16))

    #Los titulos de cada subplot
    titles = ["Objetivo", "Control", "Estado", "Adjunto"]
    #Y los valores de las funciones
    data = [zd, u, y, p]

    #Graficamos cada una de las soluciones
    for i, ax in enumerate(axs.flat):
        #Graficamos la solucion suavizada
        tcf = ax.tricontourf(x, y_coords, data[i], levels=20, cmap='viridis')
        #Configuramos la barra de colores
        fig.colorbar(tcf, ax=ax, format=ScalarFormatter(useOffset=False))
        #Ponemos el titulo y la leyenda de los ejes
        ax.set_title(titles[i])
        ax.set_xlabel('x')
        ax.set_ylabel('y')

    #Ajustamos los espacios entre subplots.
    plt.tight_layout()
    #Mostramos los graficos
    plt.show()
```

Ejecutamos el algoritmo un valor de regularización $\beta = 10^{-6}$. A continuación, se presentan las visualizaciones correspondientes a los resultados obtenidos.

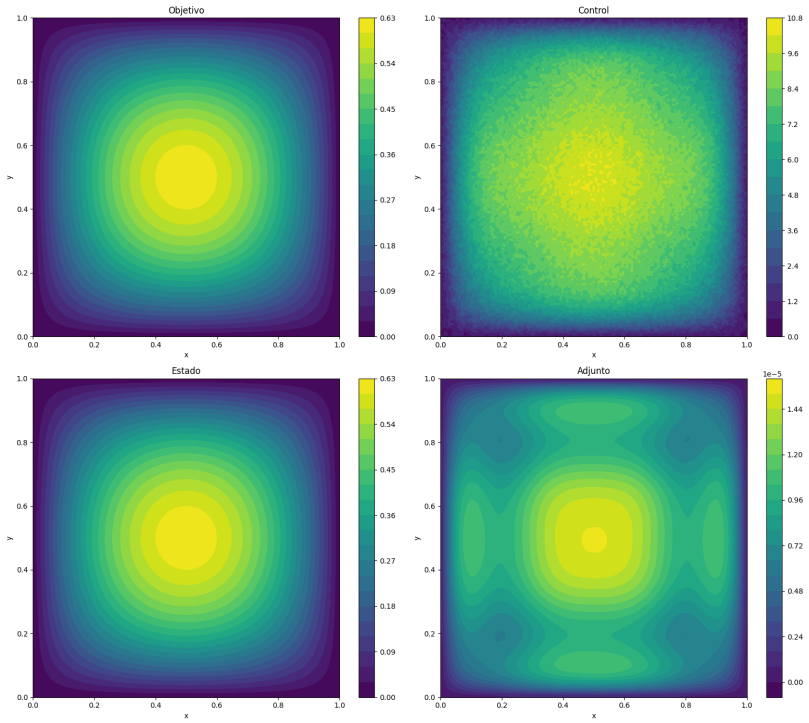

Figura 1: Visualización de las soluciones obtenidas con $\beta = 10^{-6}$.

A continuación, repetimos el experimento utilizando un valor de regularización más alto, $\beta = 10^{-3}$, con el fin de observar cómo este parámetro afecta el control óptimo y las demás variables involucradas. En las siguientes figuras se muestran los resultados correspondientes a esta nueva simulación.

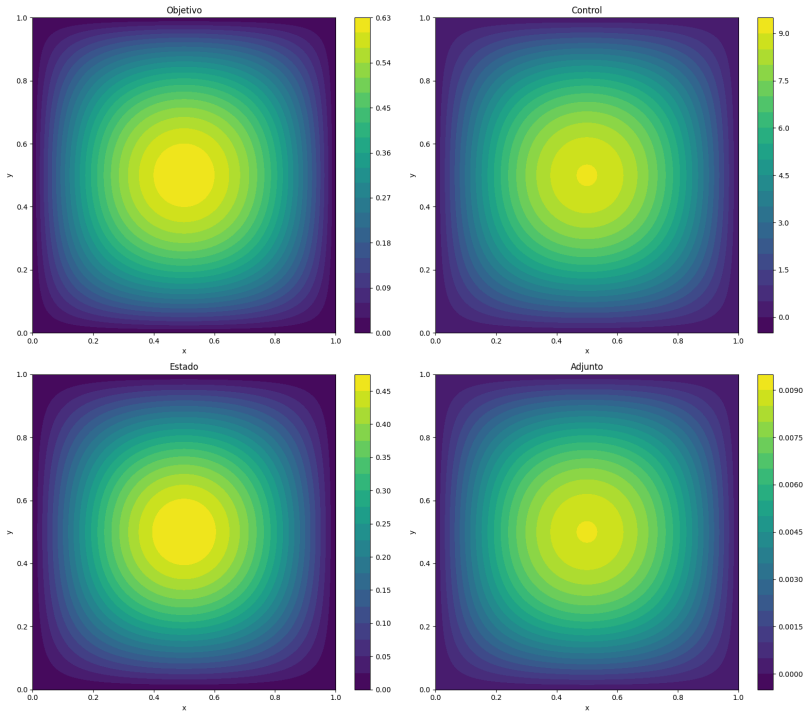

Figura 2: Visualización de las soluciones obtenidas con $\beta = 10^{-6}$.

Al observar los resultados obtenidos, notamos que la variable de estado y busca aproximarse a la función objetivo z_d, lo cual es coherente con la formulación del problema: minimizar la diferencia entre el estado alcanzado por el sistema y un perfil deseado. Cuando el parámetro de regularización β es pequeño, la solución obtenida para y se ajusta casi perfectamente a z_d. Esto ocurre porque el término de penalización sobre el control u en la función objetivo es muy débil, permitiendo que el algoritmo utilice un control más intenso para lograr parecerse al objetivo.

Sin embargo, al aumentar el valor de β, observamos que la variable de estado y ya no replica exactamente a z_d, lo cual es esperable: el término de regularización se vuelve más dominante y penaliza controles demasiado grandes. Desde el punto de vista físico, esto representa una situación en la que el uso del control está limitado por restricciones de costo o energía, por lo que el sistema no puede alcanzar exactamente el perfil deseado y se ve forzado a encontrar un equilibrio entre cercanía y eficiencia.

En conjunto, los resultados obtenidos son coherentes con la teoría y reflejan un comportamiento esperado en problemas de control óptimo. Además, se evidencia que la implementación del método de descenso de gradiente ha sido realizada correctamente, pues las soluciones numéricas responden como es debido a la variación del parámetro de regularización.

Caso con Restricciones

A continuación, realizaremos simulaciones considerando el caso restringido del problema de control óptimo, es decir, imponiendo la condición $-a \leq u \leq a$ en todo Ω. Para implementar esta restricción de manera práctica, utilizaremos una función de proyección que aplicaremos sobre u cada vez que lo modifiquemos según la dirección de mínimo descenso, asegurando que el control permanezca dentro del rango permitido.

Además, para efectos de las simulaciones, utilizaremos como función objetivo la siguiente elección:

$$z_d(x, y) = \sin(2\pi x)\sin(2\pi y),$$

la cual proporciona una referencia suave y no trivial para el estado deseado. A continuación se presenta un bloque de código que muestra la implementación de la función de proyección del control y la definición de z_d.

```
#Funcion de proyeccion de u -> -a \leq u \leq a
def Proy(a, u):
    #Si el caso es irrestricto, no hacemos nada
    if a == None:
        return u
    #Si es restricto, proyectamos cortando los valores que sobrepasen los
       limites
    return np.clip(u, -a, a)
```

Para el primer experimento, consideraremos el parámetro de regularización $\beta = 10^{-6}$ y analizaremos cómo varía la solución del problema al modificar el nivel de restricción sobre el control. En particular, vamos a graficar la solución y obtenida para distintos valores de a, específicamente $a = 20$, $a = 35$, $a = 50$, y también incluiremos el caso irrestricto (es decir, sin imposición de cotas sobre u) como referencia.

A continuación, se presentan las soluciones obtenidas para cada uno de estos casos.

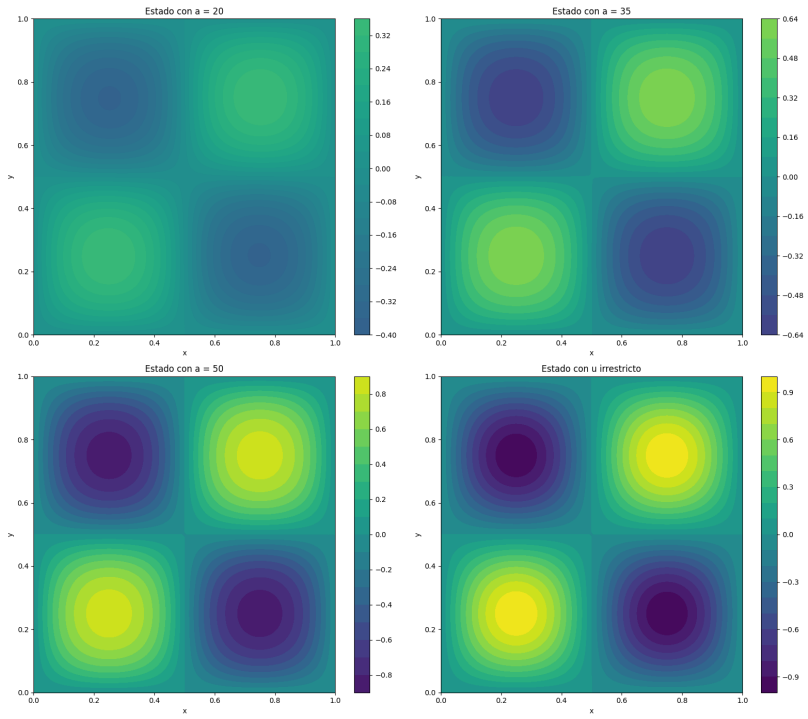

Figura 3: Visualización de $\hat{y}$ en función de a.

Las soluciones obtenidas tienen una interpretación física coherente. A pesar de que el parámetro de regularización $\beta = 10^{-6}$ es muy pequeño (lo que, en teoría, permitiría un mayor grado de libertad en la elección del control), la imposición de una restricción sobre u limita la capacidad del sistema para alcanzar el estado deseado z_d. En los casos con valores pequeños de a, el control no es lo suficientemente fuerte

como para compensar la diferencia entre el estado actual y el objetivo, por lo que la solución y intenta aproximarse a z_d, pero queda visiblemente alejada.

A medida que se flexibiliza la restricción (es decir, se incrementa el valor de a), el control tiene mayor margen de acción, lo que permite al sistema acercarse de forma más efectiva al objetivo. Finalmente, en el caso irrestricto, donde no se impone ninguna cota sobre u, la solución y prácticamente coincide con z_d, lo que confirma que el control disponible es suficiente para alcanzar exactamente el estado deseado. Esta evolución en las soluciones es consistente con el comportamiento esperado del sistema y valida la implementación numérica del modelo.

Ahora estudiaremos cómo varía la variable de control u al modificar el parámetro de regularización β, manteniendo fija la restricción sobre el control en $a = 30$. Esta exploración nos permitirá analizar el efecto que tiene β sobre la magnitud y la forma del control óptimo, dentro de un rango fijo de acción dado por a.

Para ello, realizaremos simulaciones con los siguientes valores: $\beta = 10^{-2}$, $\beta = 10^{-4}$, $\beta = 10^{-6}$ y $\beta = 10^{-8}$. A continuación, se presentan los resultados obtenidos para cada uno de estos casos.

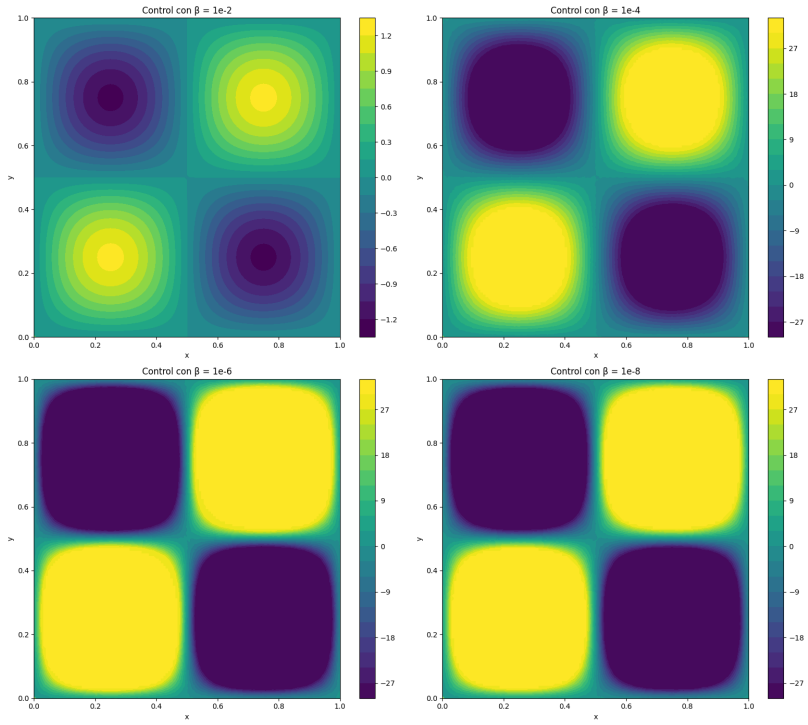

Figura 4: Visualización de $\hat{u}$ en función de β.

Los resultados obtenidos reflejan con claridad el efecto del parámetro de regularización β sobre la forma del control óptimo u. Cuando β toma valores relativamente grandes, se penaliza fuertemente el uso de controles con norma elevada, lo que induce una solución más suave y distribuida a lo largo del dominio. En estos casos, el control varía de manera continua y se mantiene dentro de un rango moderado de valores, equilibrando el compromiso entre acercar el estado y al objetivo z_d y mantener bajo el costo asociado al control.

Sin embargo, al disminuir el valor de β, dicha penalización pierde fuerza, y el modelo privilegia cada vez más minimizar el error en el estado, aun a costa de utilizar controles más intensos. Esto se manifiesta en soluciones para u que se vuelven más extremas: el control adopta esencialmente valores iguales a los extremos permitidos por la restricción, es decir, ± 30, en gran parte del dominio. Este fenómeno es ca-

racterístico de lo que se conoce como control *bang-bang*, en el cual el sistema aplica el máximo esfuerzo posible en cada punto con tal de alcanzar el estado deseado, sin preocuparse por la regularidad del control. Este comportamiento es especialmente evidente cuando β es muy pequeño, como en los casos $\beta = 10^{-6}$ y $\beta = 10^{-8}$.

Convergencia Práctica

Consideremos validar en la práctica nuestros métodos y esquemas utilizados.

Convergencia de la Discretización

Para analizar la convergencia de nuestro esquema numérico, consideraremos una solución manufacturada que satisfaga el sistema de ecuaciones que caracteriza la optimalidad del problema de control. Recordemos que este corresponde a:

$$\begin{cases} -\Delta \hat{y} = \hat{u} & \text{en } \Omega, \\ -\Delta \hat{p} = \hat{y} - z_d & \text{en } \Omega, \\ \hat{p} + \beta \hat{u} = 0 & \text{en } \Omega. \end{cases}$$

Consideramos una solución manufacturada dada por:

$$\hat{y}(x, y) = x(1 - x)y(1 - y),$$

la cual cumple condiciones de Dirichlet homogéneas en el borde del dominio. Calculamos su Laplaciano:

$$\Delta \hat{y}(x, y) = -2y(1 - y) - 2x(1 - x).$$

De la primera ecuación del sistema (1) obtenemos el control óptimo:

$$\hat{u} = -\Delta \hat{y} = 2y(1 - y) + 2x(1 - x).$$

Utilizando la tercera ecuación del sistema, hallamos el adjunto:

$$\hat{p} = -\beta \hat{u} = -2\beta y(1 - y) - 2\beta x(1 - x).$$

Finalmente, de la segunda ecuación del sistema podemos determinar z_d:

$$\begin{aligned} -\Delta \hat{p} &= \hat{y} - z_d \\ -\Delta \left[-2\beta y(1 - y) - 2\beta x(1 - x)\right] &= x(1 - x)y(1 - y) - z_d \\ 2\beta \Delta \left[y(1 - y) + x(1 - x)\right] &= x(1 - x)y(1 - y) - z_d \\ 2\beta \left[-2 + -2\right] &= x(1 - x)y(1 - y) - z_d \\ -8\beta &= x(1 - x)y(1 - y) - z_d, \end{aligned}$$

por lo que la función objetivo deseada es:

$$z_d = x(1 - x)y(1 - y) + 8\beta.$$

Para llevar a cabo el análisis de convergencia práctico, implementaremos las funciones exactas que corresponden a la solución manufacturada del problema, y luego las interpolaremos sobre la base de elementos finitos utilizada. Esto nos permitirá comparar directamente las soluciones numéricas con sus contrapartes exactas proyectadas en el espacio discreto.

```python
#Solucion manufacturada de y
def y_exact(x, y):
    return x*(1 - x)*y*(1 - y)

def Gety_exact(basis):
    x = basis.doflocs[0, :]  #Coordenadas x
    y = basis.doflocs[1, :]  #Coordenadas y
    return y_exact(x, y) #Evaluacion de y_exact en (x, y)

#Solucion manufacturada de u
def u_exact(x, y):
    return 2*y*(1-y) + 2*x*(1-x)

def Getu_exact(basis):
    x = basis.doflocs[0, :]  #Coordenadas x
    y = basis.doflocs[1, :]  #Coordenadas y
    return u_exact(x, y) #Evaluacion de u_exact en (x, y)

#Solucion manufacturada de p
def p_exact(x, y, beta):
    return -2*beta*y*(1-y) - 2*beta*x*(1-x)

def Getp_exact(basis):
    x = basis.doflocs[0, :]  #Coordenadas x
    y = basis.doflocs[1, :]  #Coordenadas y
    return p_exact(x, y) #Evaluacion de p_exact en (x, y)

#Solucion manufacturada de zd
def z_d(x, y, beta):
    return x * (1 - x) * y * (1 - y) + 8*beta
```

Con el objetivo de analizar cuantitativamente la precisión del método, hemos modificado la función `GradientMethod` para que retorne los errores cometidos respecto de las soluciones exactas correspondientes. De este modo, es posible evaluar la tasa de convergencia del esquema mediante el cálculo de las normas L^2 del error y el valor de la función objetivo. La modificación fue la siguiente

```python
#Obtenemos los valores exactos
exact_y = Gety_exact(basis)
exact_u = Getu_exact(basis)
exact_p = Getp_exact(basis)

#Obtenemos los errores L2
errory = np.linalg.norm(exact_y - y)/N
erroru = np.linalg.norm(exact_u - u)/N
errorp = np.linalg.norm(exact_p - p)/N

return errory, erroru, errorp
```

En nuestro caso, la malla utilizada se construye a partir de subdivisiones uniformes del dominio $\Omega = (0,1)^2$, lo que nos permite obtener una fórmula cerrada para el parámetro de discretización h. Como cada refinamiento biseca los triángulos existentes, se tiene que

$$h = \frac{\sqrt{2}}{2^{\text{ref}}}$$

donde ref indica el nivel de refinamiento. A continuación, iteramos sobre distintos niveles de malla refinada y registramos los errores L^2 de las soluciones aproximadas, junto con los valores correspondientes de h:

```
#Inicializamos los arreglos para los errores y h
errorsy = []
errorsu = []
errorsp = []
errorsJ = []
hs = []

#Fijamos beta
beta = 1e-3

#Recorremos distintos niveles de refinamientos
for ref in range(5, 8):
    #Calculamos h
    h = np.sqrt(2)/2**ref
    #Obtenemos los errores
    errory, erroru, errorp = GradientMethod(beta, ref = ref)

    #Y los guardamos
    errorsy.append(errory)
    errorsu.append(erroru)
    errorsp.append(errorp)
    hs.append(h)
```

A continuación, seguimos un esquema estándar para representar gráficamente la convergencia del método. Utilizaremos una escala log-log para comparar el error práctico obtenido numéricamente con una recta de referencia que representa la tasa de convergencia teórica esperada. Este tipo de representación permite visualizar de forma clara el orden de convergencia del método con respecto al parámetro h.

```
#Graficamos los errores
plt.loglog(hs, errorsy, 'o-', label="Error practico")
#Graficamos la recta teorica
plt.loglog(hs, [errorsy[1]*(h/hs[1])**1 for h in hs], 'k--', label="Error
    Teorico")
#Ponemos las leyendas
plt.xlabel("h")
plt.ylabel("Error $L^2$")
plt.title("Convergencia de y")
plt.legend()
#Mostramos
plt.show()
```

A continuación, mostramos el gráfico correspondiente al error de la variable de estado $\hat{y}$.

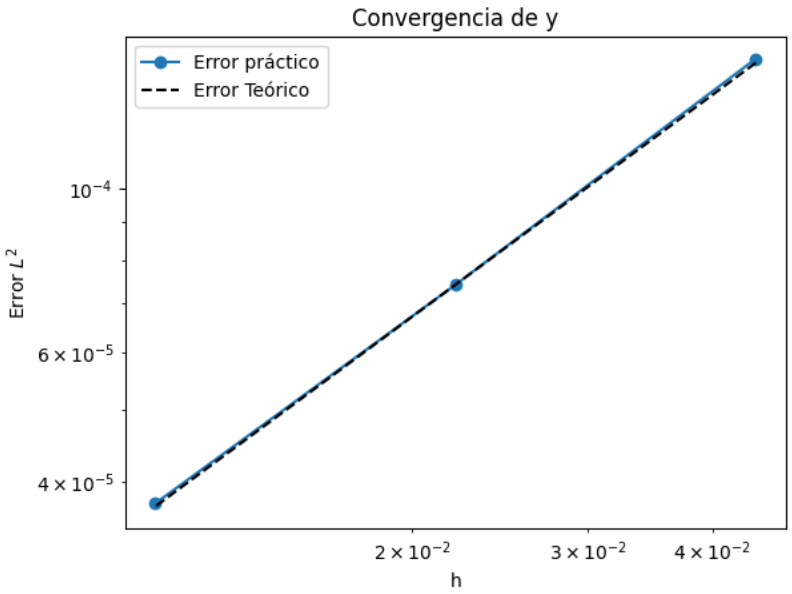

Figura 5: Convergencia de $\hat{y}$.

Mostramos ahora el gráfico correspondiente al error del control óptimo $\hat{u}$.

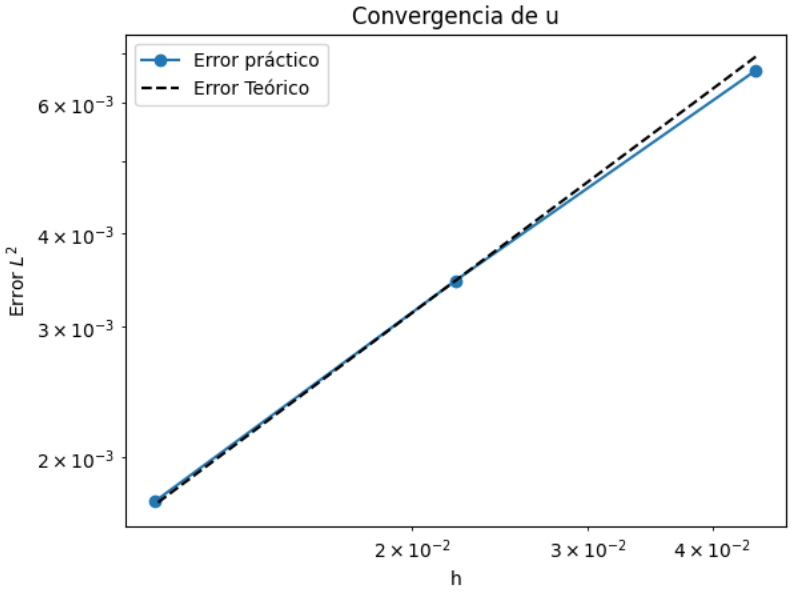

Figura 6: Convergencia de $\hat{u}$.

Finalmente, se presenta el gráfico correspondiente al error de la variable adjunta $\hat{p}$.

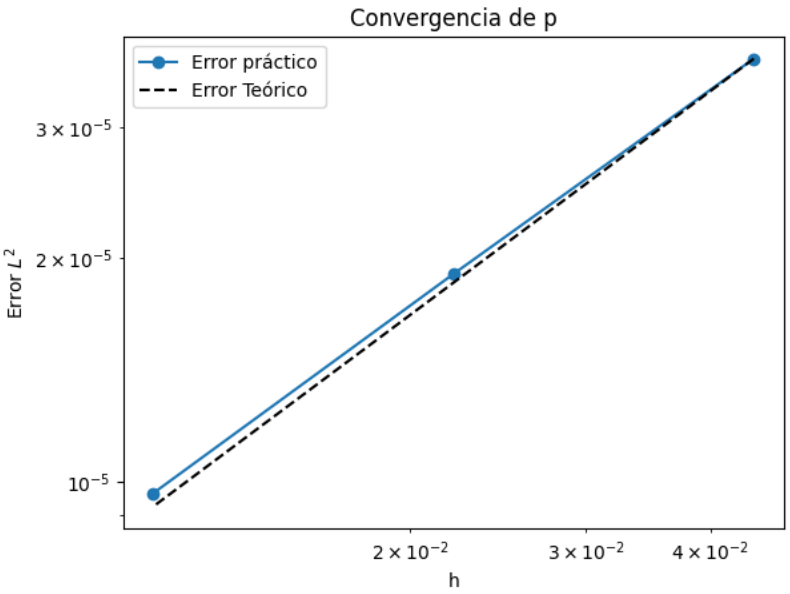

Figura 7: Convergencia de $\hat{p}$.

Estos resultados validan la teoría, ya que la convergencia observada es de orden $O(h)$, como se esperaba. La recta teórica se ajusta muy bien a los resultados prácticos obtenidos, lo que confirma la correcta implementación y efectividad del método.

Convergencia del Método del gradiente

Para estudiar la convergencia del método de optimización utilizado, se procederá a graficar dos elementos clave:

- El valor de la función objetivo J a lo largo de las iteraciones.
- La norma del gradiente $\|\nabla J\|$ también en función de las iteraciones.

Los experimentos fueron realizados con tres valores de β: $\beta = \{1e-1, 1e-2, 1e-3\}$, lo que nos permite observar cómo afecta este parámetro al comportamiento de la convergencia del método. Además, para estos experimentos, se consideró un nivel de refinamiento de 6 en la malla, lo que garantiza una resolución adecuada para estudiar el rendimiento del método. Los gráficos se muestran a continuación:

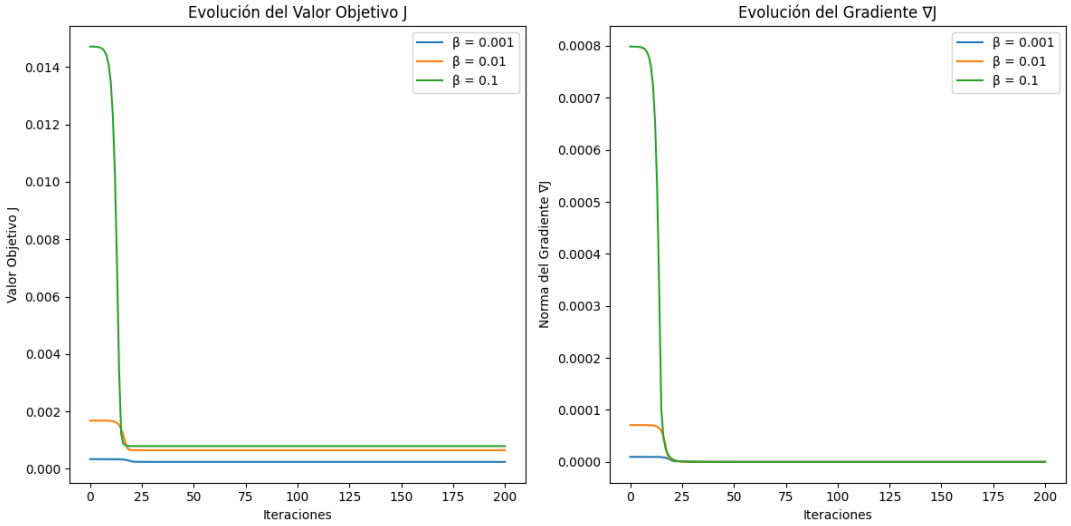

Figura 8: Convergencia del Método del Gradiente.

Los gráficos obtenidos muestran los resultados esperados, con las soluciones convergiendo hacia el óptimo. Los errores disminuyen rápidamente, lo que valida que el esquema implementado está funcionando correctamente.

Referencias

[1] Andrea Manzoni, S. Salsa y Alfio Quarteroni. *Optimal control of partial differential equations : analysis, approximation, and applications*. eng. Applied Mathematical Sciences ; v.207. Cham, Switzerland: Springer, 2021. ISBN: 3-030-77226-8.