Instituto Tecnológico Autónomo de México

Métodos Numéricos y Optimización

Práctica 1

Román Alberto Vélez Jiménez
CU: 165462

28 de septiembre de 2023

Introducción

Una función implícita es una función de la forma: $F(x, y) = 0$. Las funciones implícitas se pueden utilizar para modelar objetos geométricos. Así, un objeto geométrico puede estar definido en términos de F como $\mathcal{G} = \{(x, y) \in \mathbb{R}^2 | F(x, y) \leq 0\}$ la región del plano definida por la gráfica de F.

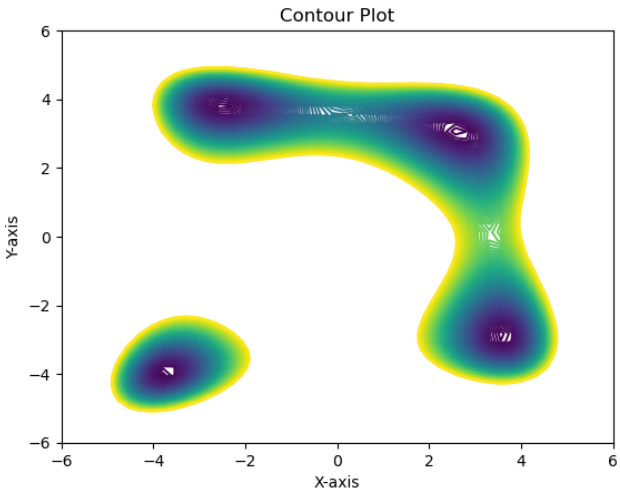


Figura 1: Curvas de nivel negativas de la Función de Himmelblau.

Problema: consideramos la región $\mathcal{G}$ determinada por $F(x, y) := (x^2 + y - 10)^2 + (x + y^2 - 12)^2 - 100$, la Función de Himmelblau [1]. A partir de esta geometría se puso a prueba métodos de integración, como Monte Carlo y Rectángulo (recursivo) en $\mathbb{R}^2$.

1. Montecarlo

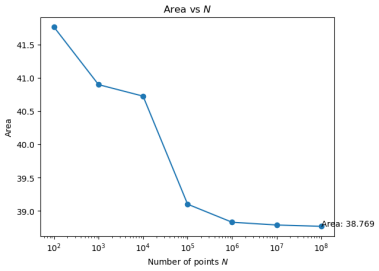
Se calculó el área de $\mathcal{G}$ usando muestras de tamaño $10^2, 10^3, \ldots, 10^8$ y obtuvieron los resultados de la TABLA. Con base en estos resultados, el área de $\mathcal{G}$ es de 38.7690 unidades.

Cuadro 1: Área de $\mathcal{G}$ para distintos tamaños de muestra.

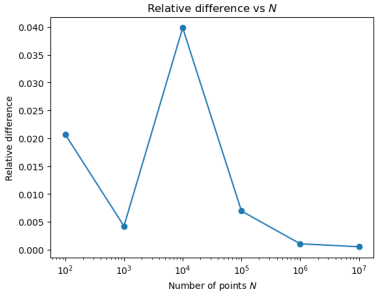
10^2	10^3	10^4	10^5	10^6	10^7	10^8
41.7600	40.8960	40.7232	39.1018	38.8303	38.7893	38.7690

En la figura 2 se observa cómo el área en términos relativos mejora (decrece) en un orden de $\mathcal{O}(1/\sqrt{n})$, como la teoría apunta [2]. En la figura 3 se observa cómo la densidad de los puntos sobre el volumen predefinido va en aumento y como con ello la estimación del área es mejor.

Por medio de este ejercicio se concluye que un mayor tamaño de muestra *siempre* mejora el área estimada. Sin embargo, la mejora es sub-lineal mientras que el tiempo de cómputo es supra-lineal, por lo que será importante evaluar la exactitud por el tiempo de cómputo esperado.

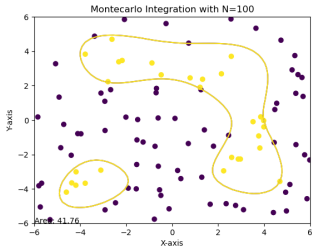


(a) Cambio del área de $\mathcal{G}$ para distintos tamaños de muestra.

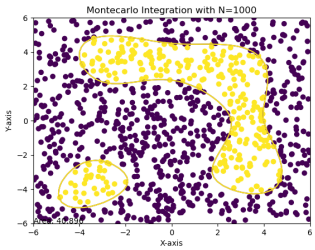


(b) Cambio relativo del área de $\mathcal{G}$ para el siguiente tamaño de muestra.

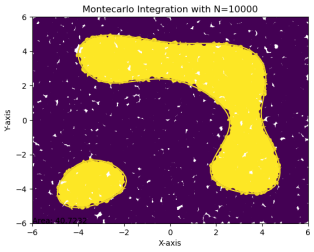
Figura 2: Área de $\mathcal{G}$ para distintos tamaños de muestra y su cambio relativo.



(a) Tamaño de muestra de 10^2



(b) Tamaño de muestra de 10^3



(c) Tamaño de muestra de 10^4

Figura 3: Integración Monte Carlo para distintos tamaños de muestra. Los puntos morados indican que los puntos están fuera del área de interés, mientras que los puntos amarillos indican que están dentro de $\mathcal{G}$.

2. Integración Recursiva

Se definió un método[1] recursivo que usa la Técnica de Integración por Rectángulo para estimar, de otra manera, el área de $\mathcal{G}$. La idea es la siguiente:

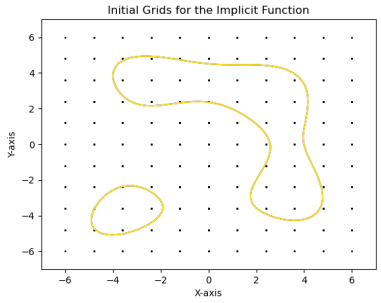
1. **Creación del enrejillado**. Se fija un área rectangular V que cubra de forma completa a $\mathcal{G}$. A partir de ello se generan h^2 rejillas sobre V, donde mayor número de rejillas h implica una mejor estimación del área. De esta forma se obtiene en enrejillado R. Ver figura 4.

2. **Evaluación de la rejilla**. Para la rejilla $r_i \in R$ se analiza si algún vértice está en el área de interés, i.e. $F(x_j, y_j) < 0, \quad$ para algún $(x_j, y_j) \in r_i$. Si todas las esquinas son positivas, i.e. $F(x_j, y_j) > 0, \quad \forall (x_j, y_j) \in r_i$, entonces esa rejilla no es de interés; en su defecto, es de interés.

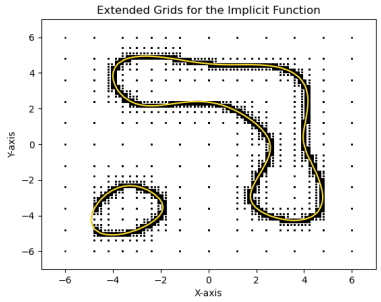
3. **Integración sobre la rejilla**. Si todos los vértices están dentro del área, entonces se añade el área de esa rejilla al área total, la cual está dada por área$(r_i) = (x_2 - x_1)(y_2 - y_1)$.

[1] El algoritmo se puede encontrar en los Anexos.

4. **Recursión**. En el caso que existan vértices dentro y fuera de $\mathcal{G}$, se hará el paso recursivo. Dicho paso consiste en romper la rejilla r_i en 2^2 subrejillas, extendiendo el enrejillado R a R^*. Si este proceso pasa más de D veces, se detiene la recursión y se añade al área total las rejillas que tengan al menos un vértice dentro de $\mathcal{G}$. Ver figura 4.



(a) Enrejillado inicial R



(b) Enrejillado extendido R^* para un nivel de recursión D.

Figura 4: Enrejillado inicial R y extendido R^* para el método de integración propuesto.

Se llevaron a cabo dos experimentos en el estudio, uno involucrando una red amplia con parámetros $h = 10$ y un alto nivel de recursión $D = 8$, mientras que el segundo experimento empleó una red más fina con $h = 50$ y un nivel de recursión más bajo de $D = 2$. Debido a que la geometría $\mathcal{G}$ en consideración es de naturaleza irregular, se logró obtener una mayor precisión en el primer experimento, donde se obtuvo un área de 38.8004, en contraste con el segundo experimento, que registró un área de 40.2984. Además, es relevante destacar que el tiempo de ejecución en ambos casos fue similar, con 0.3 segundos para el primer experimento y 0.2 segundos para el segundo, lo que refleja la misma eficiencia computacional en ambas configuraciones.

A partir de estos resultamos se concluye que los métodos recursivos también pueden competir con métodos probabilistas en cuestión de la precisión y tiempo computacional. Sin embargo, un punto importante a mencionar es que la heurística del punto 2 del método es cuestionable, porque es posible encontrar en el interior de r_i un punto $(x_j, y_j) \in$ tal que $F(x_j, y_j) < 0$. Por lo cuál podrían existir enrejillados R para geometrías $\mathcal{G}$ donde ningún vértice esté dentro del área de interés, volviendo este método menos robusto que uno probabilista.

3. Probabilidades sobre $\mathcal{G}$

En el ámbito probabilístico es común querer encontrar probabilidades sobre regiones de interés, por ejemplo calcular $\mathbb{P}[\underline{X} \in \mathcal{G}]$. Si $\underline{X} \sim \mathcal{N}_2(\underline{\mu}, \Sigma)$, donde $\mu_x = \mu_y$ y $\Sigma = 2I_{2\times 2}$, ¿cuál sería dicha probabilidad?

3.1. Integral Teórica

Teóricamente, esa probabilidad está dada por

$$\mathbb{P}[\underline{X} \in \mathcal{G}] = \int_{(x,y)\in\mathcal{G}} f(x,y)dxdy, \tag{1}$$

donde $f(x,y) = f(\underline{x}) = \frac{1}{2\pi\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(\underline{x}-\underline{\mu})^T\Sigma^{-1}(\underline{x}-\underline{\mu})\right)$. Sin embargo, para encontrar los límites de integración en $\mathcal{G} \subset \mathbb{R}^2$ se tendría que recurrir a un cambio de variable, como coordenadas polares o sobre $u := x^2 + y - 10$ y $v := x + y^2 - 12$. Aún siendo así, la forma analítica no parece tener una forma cerrada debido al Jacobeano de la integral.

3.2. Justificación

En estos contextos, resulta razonable realizar una estimación de la probabilidad $\mathbb{P}[\underline{X} \in \mathcal{G}]$ mediante el método de Monte Carlo. La elección de este enfoque se sustenta en el hecho de que al muestrear puntos de manera aleatoria en el espacio de entrada y calcular la proporción de puntos que caen dentro del área de interés respecto al total de puntos generados, se obtiene una aproximación del área gracias a la aplicación de la Ley de los Grandes Números de Borel-Cantelli. Esta ley establece que la media de una muestra aleatoria converge hacia la media de la distribución de la cual se obtiene dicha muestra. En este contexto, se procede a muestrear puntos aleatorios en el espacio de entrada y se calcula el porcentaje de puntos que se encuentran dentro de la región de interés. Formalmente, esto se expresa de la siguiente manera:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\mathcal{G}(\underline{X}_i)} = \mathbb{P}[\underline{X} \in \mathcal{G}], \quad c.s.$$

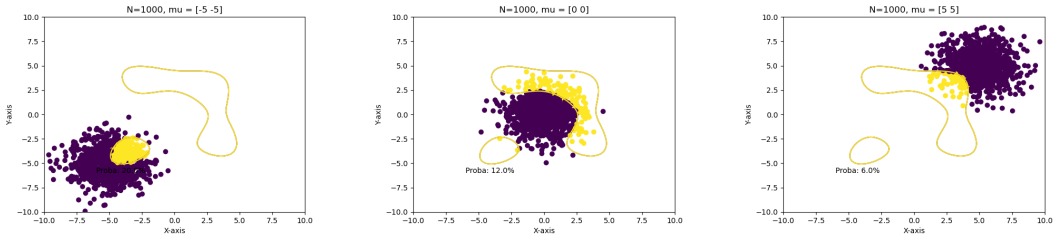
Donde $\mathbb{1}_{\mathcal{G}(\underline{X}_i)}$ denota la función indicadora que evalúa si el punto $\underline{X}_i$ se encuentra dentro de la región de interés $\mathcal{G}$, y la notación $c.s.$ indica que la convergencia es casi segura.

3.3. Experimentos

Para ejemplificar el uso de estimación de probabilidades mediante Monte Carlo, se calculó la probabilidad de la ecuación 1 para $\mu_x = \mu_y \in \{-5, -4, \ldots, 5\}$, usando muestras de tamaño $10^2, 10^3, \ldots, 10^8$. Las figura 5 ejemplifica cómo geométricamente se puede interpretar esta probabilidad mediante el Método Monte Carlo.

El cuadro 2 muestra las probabilidades estimadas para las distintas medias y tamaños de muestras. Se observa que la mayor probabilidad se encuentra en $\underline{\mu} = (2,2)^T$ y la menor probabilidad en $\underline{\mu} = -(1,1)^T$.

Este ejercicio visualiza cómo se puede usar Integración Monte Carlo para problemas comunes en la ciencia de datos. En general, este trabajo ejemplifica técnicas numéricas útiles para la estimación de probabilidades, tarea común en el día a día de un analista de datos.



(a) Probabilidad estimada para $\mu_x = \mu_y = -5$ y $N = 10^3$.

(b) Probabilidad estimada para $\mu_x = \mu_y = 0$ y $N = 10^3$.

(c) Probabilidad estimada para $\mu_x = \mu_y = 5$ y $N = 10^3$.

Figura 5: Visualización de la probabilidad estimada $\mathbb{P}[\underline{X} \in \mathcal{G}]$ para $\mu_x = \mu_y \in \{-5, 0, 5\}$ con un tamaño de muestra $N = 10^3$ mediante Integración Monte Carlo.

Cuadro 2: Probabilidades estimadas para distintas medias y tamaños de muestra

$\mu_x = \mu_y$	10^2	10^3	10^4	10^5	10^6	10^7	10^8
5	0.1000	0.0620	0.0656	0.0650	0.0661	0.0660	0.0661
4	0.2600	0.2610	0.2828	0.2806	0.2800	0.2805	0.2804
3	0.6100	0.6240	0.5806	0.5785	0.5783	0.5775	0.5778
2	0.6200	0.6120	0.6219	0.6234	0.6230	0.6222	0.6222
1	0.3800	0.3780	0.3605	0.3609	0.3590	0.3593	0.3592
0	0.1100	0.1240	0.1138	0.1169	0.1167	0.1172	0.1172
-1	0.1100	0.0590	0.0559	0.0541	0.0527	0.0525	0.0526
-2	0.1100	0.1550	0.1560	0.1520	0.1513	0.1517	0.1516
-3	0.3800	0.3300	0.3344	0.3343	0.3351	0.3349	0.3348
-4	0.3400	0.3610	0.3556	0.3571	0.3593	0.3588	0.3586
-5	0.1300	0.2010	0.1843	0.1872	0.1856	0.1860	0.1861

Anexos

El código de recursión es el siguiente:

Listing 1: Código de Integración Recursiva en Python.

```
import numpy as np

def grid_division(lp_range, h=2):
    # generate grids for a given range
    x_min, x_max, y_min, y_max = lp_range
    x_step = (x_max - x_min) / h
    y_step = (y_max - y_min) / h
    grids = []
    # generate h^2 points for the grids
    for i in range(h):
        for j in range(h):
            x1 = x_min + i * x_step
            x2 = x_min + (i + 1) * x_step
            y1 = y_min + j * y_step
            y2 = y_min + (j + 1) * y_step
            grids.append((x1, x2, y1, y2))
    return grids

def check_sign_pos(signs):
    list_signs = [sign > 0 for sign in signs]
    return all(list_signs)

def check_sign_neg(signs):
    list_signs = [sign <= 0 for sign in signs]
    return all(list_signs)

def integrate_grid(list_sign, grid):
    if check_sign_pos(list_sign):
        # outside
        return 0.0
    else:
        # heuristic
        x1, x2, y1, y2 = grid
        return (x2 - x1) * (y2 - y1)

def unpack_grid(grid):
    x1, x2, y1, y2 = grid
    return [(x1, y1), (x1, y2), (x2, y1), (x2, y2)]

def integrate_recursive_armonic(F, grids, depth):
    # global variable to save the extended grids
    global grid_extended
```

```
    # finished recursion
    if len(grids) == 0:
        return 0.0

    # max recursion --> integrate
    if depth == 0:
        area = 0.0
        for grid in grids:
            points = unpack_grid(grid)
            list_sign = [np.sign(F(x, y)) for x, y in points]
            area += integrate_grid(list_sign, grid)
        return area

    # recursive
    grid = grids.pop()
    # get points
    points = unpack_grid(grid)
    # check sign of points
    list_sign = [np.sign(F(x, y)) for x, y in points]
    # integrate
    if check_sign_pos(list_sign) or check_sign_neg(list_sign):
        return integrate_grid(list_sign, grid) +
            integrate_recursive_armonic(F, grids, depth)
    else:
        new_grids = grid_division(grid, h=2)
        # save to grid extended
        grid_extended.extend(new_grids)
        return integrate_recursive_armonic(F, new_grids, depth-1) +
            integrate_recursive_armonic(F, grids, depth)
```

Referencias

[1] Xin-She Yang. *Nature-inspired optimization algorithms.* Academic Press, 2020.

[2] Erick Palacios Moreno. *Optimización.* Jupyter Notebook, 2021.