FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Entrega Teórica 1 - Análisis Numérico I

[75.12] Análisis Numérico I
Curso Tarela
Primer cuatrimestre de 2021

Alumno 1:	Argel, Ignacio
Número de padrón:	104351
Email:	iargel@fi.uba.ar
Alumno 2:	Bareiro, Facundo
Número de padrón:	103807
Email:	fbareiro@fi.uba.ar

Índice

1. Introducción

Los objetivos de este informe constan en analizar en profundidad la resolución de los problemas de sistemas de ecuaciones lineales mediante el método SOR, estudiar para este caso particular la Matriz de Hilbert, desarrollar un programa que permita determinar experimentalmente el factor de sobre relajación óptimo (ω) para los datos que otorga el enunciado y mediante los datos obtenidos poder sacar conclusiones que respalden la teoría vista en el curso.
Para este se utilizo Python como lenguaje de programación y el siguiente conjunto de librerías:

- NumPy : Para trabajar con matrices, obtener las normas infinito.
- SimPy: Para calcular las integrales en el termino independiente.
- Matplotlib: Para generar los gráficos utilizados en este informe.

2. Desarrollo

2.1. Obtener Matriz de Hilbert

Para llegar a la matriz de Hilbert se tuvo en cuenta la expresión del enunciado:

$$H_{j+1,k+1} = (\frac{b^{j+k+1}-a^{j+k+1}}{j+k+1}) \text{ con } j,k = 0,...,N$$

mediante Python con dos bucles for y respaldados por la expresión de arriba se llegó a la siguiente matriz de 10x10.
Observación: Redujimos los dígitos mostrados en pantalla para permitir la visualización. El cálculo está hecho sobre los 16 decimales por defecto de simple precisión en Python.

$$\begin{bmatrix}
1{,}088 & 0{,}591 & 0{,}429 & 0{,}350 & 0{,}305 & 0{,}276 & 0{,}257 & 0{,}245 & 0{,}237 & 0{,}232 \\
0{,}591 & 0{,}429 & 0{,}350 & 0{,}305 & 0{,}276 & 0{,}257 & 0{,}245 & 0{,}237 & 0{,}232 & 0{,}230 \\
0{,}429 & 0{,}350 & 0{,}305 & 0{,}276 & 0{,}257 & 0{,}245 & 0{,}237 & 0{,}232 & 0{,}230 & 0{,}229 \\
0{,}350 & 0{,}305 & 0{,}276 & 0{,}257 & 0{,}245 & 0{,}237 & 0{,}232 & 0{,}230 & 0{,}229 & 0{,}230 \\
0{,}305 & 0{,}276 & 0{,}257 & 0{,}245 & 0{,}237 & 0{,}232 & 0{,}230 & 0{,}229 & 0{,}230 & 0{,}232 \\
0{,}276 & 0{,}257 & 0{,}245 & 0{,}237 & 0{,}232 & 0{,}230 & 0{,}229 & 0{,}230 & 0{,}232 & 0{,}236 \\
0{,}257 & 0{,}245 & 0{,}237 & 0{,}232 & 0{,}230 & 0{,}229 & 0{,}230 & 0{,}232 & 0{,}236 & 0{,}241 \\
0{,}245 & 0{,}237 & 0{,}232 & 0{,}230 & 0{,}229 & 0{,}230 & 0{,}232 & 0{,}236 & 0{,}241 & 0{,}247 \\
0{,}237 & 0{,}232 & 0{,}230 & 0{,}229 & 0{,}230 & 0{,}232 & 0{,}236 & 0{,}241 & 0{,}247 & 0{,}253 \\
0{,}232 & 0{,}230 & 0{,}229 & 0{,}230 & 0{,}232 & 0{,}236 & 0{,}241 & 0{,}247 & 0{,}253 & 0{,}261
\end{bmatrix}$$

2.2. Obtención del término independiente

En este caso se presentaron dos tipos de formas de hallar los valores del término independiente. La primera sería llegar mediante la resolución analítica de la integral a una expresión que dependa de las constantes a y b; La segunda, utilizada finalmente para la resolución es mediante la librería SimPy resolver la integral en cuestión sin ningún paso intermedio.

$$c_{j+1} = \int_a^b -4x(x-1)x^j dx$$

Finalmente se obtuvo el siguiente vector:

$$\begin{bmatrix} 0{,}650243217488076 \\ 0{,}315938924200197 \\ 0{,}181570231662825 \\ 0{,}113799256785772 \\ 0{,}0745258771503912 \\ 0{,}0494591342881323 \\ 0{,}0322441905633196 \\ 0{,}0197003950628422 \\ 0{,}0100896502469084 \\ 0{,}00239487716254494 \end{bmatrix}$$

2.3. Obtención de L, U y D

En el caso de este ejercicio para obtener L, U y D y luego llegar a la solución $x^{(k)}$, hay que basarse en la matriz de Hilbert. sabiendo que L es la matriz triangular inferior con los coeficientes cambiados de signo, con dos bucles for e imponiendo la condición de que donde filas <= columnas se cambie el valor a cero, se obtiene L. Para U es lo mismo solo que filas >= columnas ya que es la matriz triangular superior y para D como solo conserva los elementos de la diagonal, es el mismo proceso pero con filas ! =columnas.

3. Marco teórico del método SOR

3.1. El método

Para poder resolver el método SOR se decidió optar por la forma matricial del mismo, el cual se detallará a continuación:

Se tiene que cada solución para un k paso de iteración, puede averiguarse mediante la siguiente expresión:

$$x^{(k)} = T_\omega x^{(k-1)} + c_\omega \tag{1}$$

siendo T y C constantes que dependen del método iterativo utilizado. Para el método SOR, en la teórica se llega a una expresión analítica la cual es:

$$x_i^{(k)} = (1-\omega)x_i^{(k-1)} + \frac{\omega}{a_{i,i}}\{b_i - \sum_{j=1}^{i-1} a_{i,j}x_j^{(k)} - \sum_{j=i+1}^{n} a_{i,j}x_j^{(k-1)}\} \tag{2}$$

donde los términos con a son los elementos de una matriz A genérica (en nuestro caso la de Hilbert). Para relacionar la expresión con el método de Gauss Seidel, se la expresa también como:

$$x_i^{(k)} = (1-\omega)x_i^{(k-1)} + \omega[x_i^{(k)}]_{GS} \tag{3}$$

donde $[x_i^{(k)}]_{GS}$ se obtiene como:

$$x_i^{(k)} = -\frac{1}{a_{i,i}} \sum_{j=1}^{i-1} a_{i,j} x_j^{(k)} - \frac{1}{a_{i,i}} \sum_{j=i+1}^{n} a_{i,j} x_j^{(k-1)} + \frac{b_i}{a_{i,i}} \tag{4}$$

y donde ω es el parámetro de relajación, que para asegurar la convergencia del método debe ser: $0 < \omega < 2$ aunque a fines practicos en este curso lo tomamos $1 < \omega < 2$. Una vez que se tiene la expresión 2), para poder llevarla a la forma de la expresión 1), se multiplica la matriz por $a_{i,i}$, se agrupan los terminos según el paso de iteración y finalmente queda la expresión de esta manera:

$$a_{i,i} x_i^{(k)} + \omega \sum_{j=1}^{i-1} a_{i,j} x_j^{(k)} = (1-\omega) x_i^{(k-1)} a_{i,i} x_i^{(k-1)} - \omega \sum_{j=i+1}^{n} a_{i,j} x_j^{(k-1)} + \omega b_i \tag{5}$$

que si tenemos en cuenta que la matriz A es:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix}$$

y que se puede descomponer como:

$$D - L - U$$

$$\begin{bmatrix} a_{1,1} & 0 & \cdots & 0 \\ 0 & a_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n,n} \end{bmatrix} - \begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{2,1} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & 0 \end{bmatrix} - \begin{bmatrix} 0 & a_{1,2} & \cdots & a_{1,n} \\ 0 & 0 & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

volviendo a la expresión 5), la expresamos de la siguiente manera:

$$Dx^k - \omega L x^{(k)} = (1-\omega) D x^{(k-1)} + \omega U x^{(k-1)} + \omega b \tag{6}$$

y de aquí entonces podemos obtener T y C basándonos en la expresión 1)

$$T_\omega = (D - \omega L)^{-1} [(1-\omega) D + \omega U] \tag{7}$$

$$c_\omega = \omega (D - \omega L)^{-1} b \tag{8}$$

Con L, U y D definidas como en la sección anterior, puede resolverse el método con las expresiones 1), 7) y 8), es decir definiendo T_ω y C_ω y luego reemplazandolas en 1).

3.2. Criterio de parada

Una vez que se conoce como encontrar el $x^{(k)}$, se define un w entre 1 y 2, se define una semilla ($x^{(0)}$) para el primer paso de iteración (que suele ser el vector nulo) y una vez encontrado el $x^{(1)}$, se procede a hallar $x^{(2)}$ y así se entra en el ciclo iterativo. Sin embargo, el mismo debe tener un criterio de convergencia que defina cuándo el ciclo debe terminar. Este criterio se calcula como:

$$\frac{\|x^{(k)} - x^{(k-1)}\|}{\|x^{(k)}\|} < TOL$$

donde en ambos casos se toma la norma infinita y donde TOL se refiere a la tolerancia, la cual la define el ejercicio (En nuestro caso era de $10^{(-4)}$). Por lo tanto una vez que se alcanza esta tolerancia, el ciclo termina y la solución es la del último paso de iteración.

4. Resultados

4.1. Obtención experimental del factor de relajación

Como fue mencionado anteriormente, el valor de ω utilizado en los ejercicios se encuentra entre 1 y 2. Sin embargo, (como es el caso de este ejercicio) el ω muchas veces no es dato. Por eso la pregunta es ¿Cuál se elige?. Existe para estos ejercicios lo que se denomina ω_{Optimo} y es el ω que hará que el método SOR converge más rápidamente, es decir en la menor cantidad de iteraciones, y por lo tanto el proceso se mejore. Si la matriz del ejercicio es definida positiva y tridiagonal, el ω_{Optimo} se calcula como:

$$\omega_{Optimo} = \frac{2}{1 + \sqrt{1 - \rho(T_j)^2}}$$

Sin embargo, en este ejercicio la matriz de Hilbert no cumple estas condiciones y es por eso que debe hallarse experimentalmente de la siguiente manera: Para la misma matriz, el mismo término independiente, la misma tolerancia, la misma semilla y misma precisión se hace variar el valor de w entre 1 y 2. De esta manera, para cada w el proceso terminará en cierta cantidad de iteraciones, y el ω_{Optimo} será aquel que traiga consigo la menor cantidad de iteraciones.
Semilla a utilizar:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

Se muestra a continuación el gráfico que muestra la cantidad de iteraciones para cada ω:

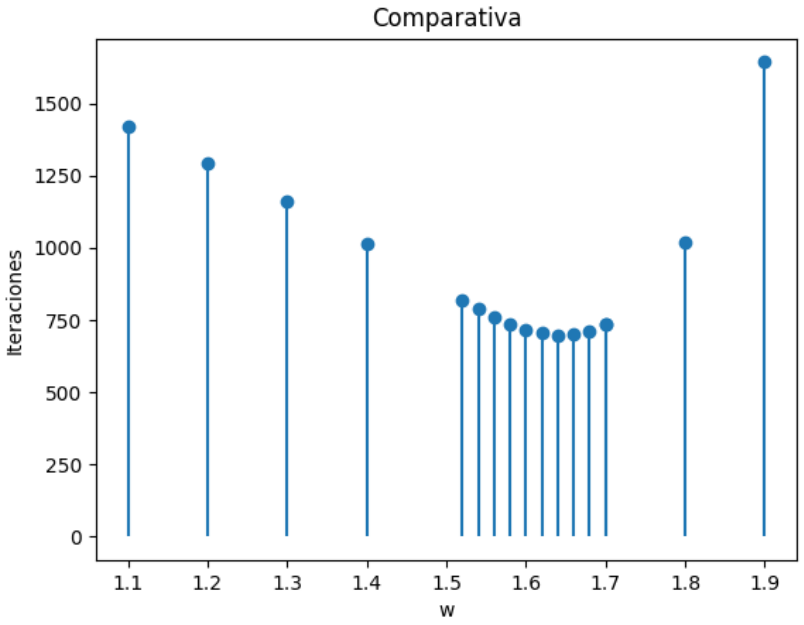

Figura 1: Grafica comparativa de iteraciones dependientes del factor de relajación .

4.2. Soluciones dependientes del factor de relajación

A su vez se adjuntan las distintas soluciones que se obtuvieron al variar ω:
(Para la visualización se utilizo redondeo truncado en 4 decimales, pero los calculos fueron reali-

zados en 16 digitos(simple precisión))

ω	Resultado
1.1	$[0{,}0216 \quad 3{,}7182 \quad -3{,}1509 \quad -0{,}7291 \quad -0{,}0965 \quad 0{,}1167 \quad 0{,}1171 \quad 0{,}0591 \quad -0{,}0036 \quad -0{,}0537]^T$
1.2	$[0{,}0170 \quad 3{,}7916 \quad -3{,}4321 \quad -0{,}3708 \quad -0{,}1866 \quad 0{,}0781 \quad 0{,}0802 \quad 0{,}0488 \quad 0{,}0045 \quad -0{,}0322]^T$
1.3	$[0{,}0127 \quad 3{,}8643 \quad -3{,}7280 \quad 0{,}0540 \quad -0{,}3720 \quad 0{,}0868 \quad 0{,}0359 \quad 0{,}0468 \quad 0{,}0109 \quad -0{,}0134]^T$
1.4	$[0{,}0088 \quad 3{,}9343 \quad -4{,}0345 \quad 0{,}5495 \quad -0{,}6760 \quad 0{,}1800 \quad -0{,}0457 \quad 0{,}0666 \quad 0{,}0078 \quad 0{,}0068]^T$
1.52	$[0{,}0045 \quad 4{,}0180 \quad -4{,}4282 \quad 1{,}2546 \quad -1{,}2313 \quad 0{,}4746 \quad -0{,}2671 \quad 0{,}1689 \quad -0{,}0482 \quad 0{,}0511]^T$
1.54	$[0{,}0037 \quad 4{,}0331 \quad -4{,}4999 \quad 1{,}3874 \quad -1{,}34756 \quad 0{,}5508 \quad -0{,}3260 \quad 0{,}2013 \quad -0{,}0701 \quad 0{,}0639]^T$
1.56	$[0{,}0029 \quad 4{,}0485 \quad -4{,}5741 \quad 1{,}5257 \quad -1{,}4714 \quad 0{,}6365 \quad -0{,}3934 \quad 0{,}2400 \quad -0{,}0978 \quad 0{,}07948]^T$
1.58	$[0{,}0021 \quad 4{,}0646 \quad -4{,}6513 \quad 1{,}6701 \quad -1{,}6033 \quad 0{,}7326 \quad -0{,}4706 \quad 0{,}2861 \quad -0{,}1325 \quad 0{,}0984]^T$
1.6	$[0{,}0012 \quad 4{,}0815 \quad -4{,}7325 \quad 1{,}8216 \quad -1{,}7440 \quad 0{,}8404 \quad -0{,}5587 \quad 0{,}3406 \quad -0{,}1758 \quad 0{,}1216]^T$
1.62	$[0{,}0004 \quad 4{,}0993 \quad -4{,}8178 \quad 1{,}9808 \quad -1{,}8939 \quad 0{,}9607 \quad -0{,}6592 \quad 0{,}4048 \quad -0{,}2292 \quad 0{,}1499]^T$
1.64	$[-0{,}0005 \quad 4{,}1183 \quad -4{,}9085 \quad 2{,}1491 \quad -2{,}0539 \quad 1{,}0951 \quad -0{,}7738 \quad 0{,}4801 \quad -0{,}2949 \quad 0{,}1846]^T$
1.66	$[-0{,}0015 \quad 4{,}1389 \quad -5{,}0058 \quad 2{,}3280 \quad -2{,}2250 \quad 1{,}2452 \quad -0{,}9045 \quad 0{,}5683 \quad -0{,}3753 \quad 0{,}2269]^T$
1.68	$[-0{,}0026 \quad 4{,}1612 \quad -5{,}1111 \quad 2{,}5193 \quad -2{,}4082 \quad 1{,}4130 \quad -1{,}0534 \quad 0{,}6713 \quad -0{,}4734 \quad 0{,}2786]^T$
1.7	$[-0{,}0038 \quad 4{,}1856 \quad -5{,}2254 \quad 2{,}7247 \quad -2{,}6049 \quad 1{,}6007 \quad -1{,}2233 \quad 0{,}7915 \quad -0{,}5926 \quad 0{,}3418]^T$
1.8	$[-0{,}0129 \quad 4{,}3654 \quad -6{,}0404 \quad 4{,}1029 \quad -3{,}8801 \quad 2{,}9757 \quad -2{,}5391 \quad 1{,}7778 \quad -1{,}7042 \quad 0{,}9451]^T$
1.9	$[-0{,}0352 \quad 4{,}8029 \quad -7{,}9218 \quad 6{,}9563 \quad -6{,}2510 \quad 5{,}9887 \quad -5{,}6033 \quad 4{,}2100 \quad -4{,}9373 \quad 2{,}7691]^T$

5. Conclusión

En primera instancia pudimos observar que el ω que genera la menor cantidad de iteraciones se encontraba entre 1,6 y 1,7. Por lo tanto generando un salto de ω más pequeño en este intervalo, concluimos que el ω_{Optimo} es aproximadamente 1,64 y la cantidad de iteraciones que genera es de 697. También podemos ver que se mejora claramente respecto de $\omega = 1$, correspondiente al método de Gauss-Seidel. A pesar de ser un número elevado de iteraciones en comparación con los ejercicios que se presentaron en el curso, creemos que es adecuado el valor teniendo en cuenta que la matriz dada no presenta ceros, y tiene una dimensión apreciable. Por eso nos mostramos conformes con los resultados que arroja el programa.

6. Anexo

```
import numpy as np
import matplotlib.pyplot as plt
from sympy import *
def sorHilbert(N,w,semilla):
    'constantes a usar en el problema'
    a = 0
    b = 0.05 + 103807*(10**(-5))
    'Defino la matriz de hilbert'
    arreglos =[]
    for j in range(N):
        subarreglo = [0 for i in range(N)]
        for k in range(N):
            constanteHilbert = j+k+1
            subarreglo[k] = (b**(constanteHilbert) - a**(constanteHilbert))/
                constanteHilbert
        arreglos.append(subarreglo)
    hilbertMatrix = np.array(arreglos)

    'Defino el termino independiente'
    x= Symbol('x')
    arregloIndep = []
    for j in range(N):
        fx = -4*x*(x-1)*(x**j)
        res = Integral(fx, (x, a, b)).doit()
        arregloIndep.append(res)
    c = np.array(arregloIndep)

    'Defino L, U, D'
    U = hilbertMatrix.copy()
    for fil in range(N):
        for col in range(N):
            if (col <= fil) :
                U[fil,col] = 0
    L = hilbertMatrix.copy()
    for fil in range(N):
        for col in range(N):
            if (fil <= col) :
                L[fil,col] = 0
    D = hilbertMatrix.copy()
    for fil in range(N):
        for col in range(N):
            if (fil != col) :
                D[fil,col] = 0
    U = U * -1
    L = L * -1

    'Calculo Tw'
    Tw = np.linalg.inv(D - w*L) @ ((1-w) * D+ w * U)

    'Calculo Cw'
    Cw = (w * np.linalg.inv(D - w*L)) @ c

    'Calculo SOR'
    iteraciones = 0
    parada = 10 ** (-4)
    tol = 1
    x_k = semilla
    while((tol > parada)):
        x_kmenos1 = x_k.copy()
        x_k = Tw @ x_k + Cw
        diferencia = x_k - x_kmenos1
        iteraciones +=1
        tol = (np.linalg.norm(diferencia, np.inf))/(np.linalg.norm(x_k, np.inf))
```