Proyecto 1
Multigrid

[Camilo Valenzuela Carrasco]

15 de mayo de 2017

1. Introducción

En el ámbito de la ciencia aplicada es múy común modelar procesos físicos utilizando ecuaciones matemáticas que capturan el comportamiento de dicho proceso.

Un caso particual de estos modelos matemáticos son las ecuaciones diferenciales parciales (EDP) donde se busca una función matemática que cumpla dicha ecuación. En algunas ocasiones se puede encontrar la solución que cumple la ecuación de forma algebráica, pero existen EDP que no se pueden resolver de forma analítica por lo que se desarrollaron métodos para poder llegar a una solución aproximada de forma numérica.

La finalidad de este trabajo es analizar y comparar tres algoritmos iterativos que buscan resolver Ecuaciones Diferenciales Parciales de forma numérica. Se comenzará realizando una pequeña descripción de cada algoritmo, luego se describirá el problema que se intenta resolver, la implementación realizada y los resultados obtenidos, para finalizar con las conclusiones realizadas a partir de los resultados obtenidos.

2. Discretización

Una Ecuación Diferencial Parcial (EDP) generalmente está definida en un dominio continuo, dependiendo del tipo de EDP este dominio puede estar acotado en un intervalo $[a, b]$ o no acotado.

Las EDP con dominio acotado se denominan Boundary Value Problem (BVP) o Problema de Valor de Frontera, este dominio acotado se puede dividir en N puntos donde la distancia entre los puntos es:

$$|x_n - x_{n+1}| = h = \frac{b-a}{N} \tag{1}$$

Además en este tipo de ecuaciones se puede discretizar las derivadas parciales de la ecuación utilizando una técnica llama *diferencias finitas* que se basa en una expansión de taylor para poder aproximar dichas derivadas. Un ejemplo de una derivada parcial utilizando discretización de segundo orden queda de la forma

$$\frac{\delta^2 u}{\delta x^2} = \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} \tag{2}$$

$$\text{Con: } u(a) = c_1, u(b) = c_2 \tag{3}$$

Dada un BVP de la forma

$$\frac{\delta^2 u}{\delta x^2} = f(x) \tag{4}$$

Se puede discretizar de la forma

$$U_{i-1} - 2U_i + U_{i+1} = h^2 f(x_i), \text{ con } i = 1 \dots N \tag{5}$$

Que de forma matricial queda

$$\begin{bmatrix} -2 & 1 & 0 & 0\dots & 0 \\ 1 & -2 & 1 & 0\dots & 0 \\ \ddots & \ddots & \ddots & 1 & -2 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} = h^2 \begin{bmatrix} f(x_1) - c_1 \\ f(x_2) \\ \vdots \\ f(x_N) - c_2 \end{bmatrix} \tag{6}$$

Y se obtiene un sistema de ecuaciones lineales de la forma

$$Au = f \tag{7}$$

3. Gauss Seidel

El método de Gauss Seidel [1] es un algoritmo iterativo que se utiliza para resolver sistema de ecuaciones lineales de la forma $Au = f$. La solución que entrega este algoritmo depende de un initial guess que se le entrega en la primera iteración, si el sistema cuenta con varias soluciones el método va a tender a encontrar la más cercana a ese initial guess.

Una propiedad importante de este algoritmo es que si la matriz es diagonal dominante ($a_{ii} > \sum_{j \neq i} a_{ji}$), este método asegura convergencia.

El algoritmo puede ser descrito de la siguiente forma:

1. Sea v_0 un initial guess, v_n la solución aproximada en la iteración n, A una matriz cuadrada que se descompone de la forma $A = L + D + U$, Donde L es una matriz triangular inferior, D es una matriz diagonal y U una matriz triangular superior, además la diagonal de D y U es igual a 0.

2. El sistema $Av = f$ puede ser escrito de la forma

$$(L + D + U)v = f \tag{8}$$
$$(L + D)v + Uv = f \tag{9}$$
$$(L + D)v = f - Uv \tag{10}$$
$$v_{n+1} = (L + D)^{-1}(f - Uv_n) \tag{11}$$

A grandes rasgos este algoritmo comienza con un Initial Guess y en cada iteración va ajustando esta solución hasta llegar a un máximo de iteraciones o una toleracia mínima.

En el caso de buscar una tolerancia mínima se puede utilizar el residuo $||r^n||_2 = ||Av^n - f||_2$ que nos entrega una aprixmación del error $||e||_2 = ||u - v||_2$ donde u es la solución exacta y v es la aproximada.

4. V-cycle Scheme

V-Cycle Scheme [1] es un algoritmo que utiliza como base la relajación Gauss Seidel, pero en cada iteración de V-Cycle trata de corregir el error resolviendo problemas más pequeños y así buscando una convergencia más rápida.

Este algoritmo puede ser definido de forma recursiva, dado ν_1 y ν_2, un initial guess v_0 y una discretización del dominio en N nodos con $h = \frac{|b-a|}{N}$

1. Sea $v^h \to v - cycle^h(v^h, f^h)$

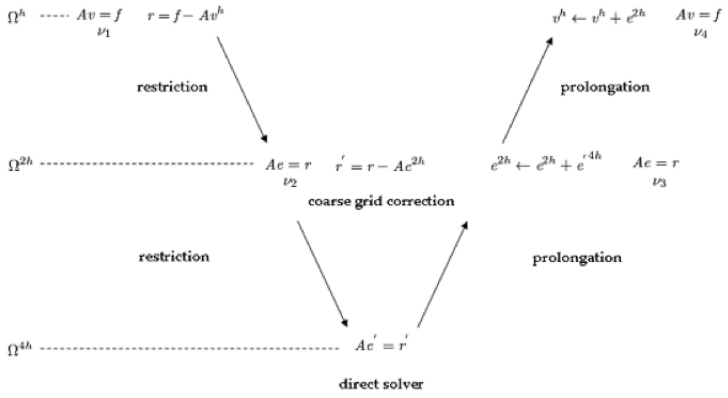


Figura 1: V-Cycle Scheme

2. Se realizan ν_1 iteraciones o relajaciones Gauss Seidel a nuestra solución actual v^h.
3. Se calcula
$$r^h = f^h - A^h v^h \tag{12}$$
4. Se restringe el residuo para llevarlo a una malla más gruesa
$$r^{2h} = I_h^{2h} r^h \tag{13}$$
Esto deja r^{2h} con $\frac{N}{2^d}$ nodos, con d la dimensión del dominio.
5. Se realiza la llamada recursiva $e^{2h} \rightarrow v - cycle e^{2h}(0, r^{2h})$
6. Luego se interpola el error, volviendolo a una malla más fina
$$e^h = I_{2h}^h e^{2h} \tag{14}$$
7. Se corrige la solución actual $x^h = x^h + e^h$
8. Finalmente se realizan ν_2 relajaciones Gauss Seidel.

Al igual que en el método Gauss Seidel, V-Cycle se puede detener luego de cierta cantidad de iteraciones o con un umbral de la norma del residuo.

5. Full Multigrid

El algoritmo Full Multigrid (FMG) utiliza una idea parecida a V-cycle, a diferencia del algoritmo anterior, FMG trata de resolver siempre el mismo problema pero a distintas escalas comenzando por la escala más gruesa y luego utilizando la solución obtenida como initial guess en la malla más gruesa así buscando una convergencia mucho más rápida.

Al igual que V-cycle este algoritmo se puede describir de forma recursiva, sean $\nu_0, \nu_1, \nu_2 \in \mathcal{N}$ FMG se define de la siguiente forma

1. Sea $v^h \rightarrow FMG^h(f^h)$
2. Si estamos en la malla más gruesa, $v^h \rightarrow 0$ se pasa directamente al paso 4.
Si no
$$f^{2h} \rightarrow I_h^{2h} f^h \tag{15}$$
$$v^{2h} \rightarrow FMG^{2h}(f^{2h}) \tag{16}$$
$$\tag{17}$$

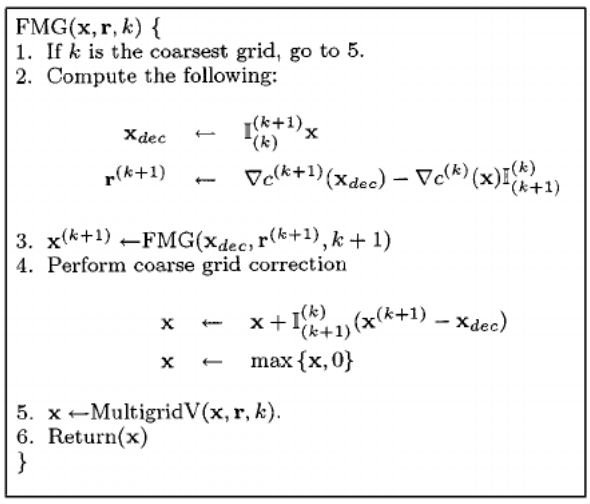


Figura 2: Full Multigrid Residuo.

3. Corregir $v^h \rightarrow I_{2h}^h v^{2h}$
4. Se realiza $v^h \rightarrow v - cycle^h(v^h, f^h)$ ν_0 veces, con los parámetros ν_1, ν_2 para $v - cycle^h$

A diferencia de los algoritmos anteriores FMG es sólo una iteración, por lo que se necesita probar distintos parámetros de ν_0, ν_1, ν_2 para encontrar la mejor combinación de ellos que entreguen el menor residuo.

Como Full Multigrid es un esquema como V-Cycle existen otros algoritmos parecidos, se implementó el FMG propuesto por [2] que se detalla en la figura 2.

A diferencia del primer esquema FMG, en este caso se utiliza el residuo y la solución actual se va actualizando según el residuo de las mallas más gruesas (muy parecido a un esquema v-cycle), en este caso cuando se cambia de malla se resuelve un problema $Ae = r$, no como en el esquema normal de FMG que se resuelve siempre $Av = f$ en cada malla.

6. Problema a resolver

Se tiene la siguiente ecuación de poisson

$$\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2} = -2\left[(1-6x^2)y^2(1-y^2) + (1-6y^2)x^2(1-x^2)\right] = f(x,y) \tag{18}$$

$$u = 0 \text{ en } x = 0, x = 1, y = 0, y = 1 \tag{19}$$

$$0 \leq x, y \leq 1 \tag{20}$$

Se necesita encontrar una solución numérica para dicha ecuación utilizando los métodos previamente detallados.

6.1. Discretización

Para poder resolver la ecuación de forma numérica se realizó una discretizacion de segundo orden de la forma

$$u_{xx} = \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2}, u_{yy} = \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h^2} \tag{21}$$

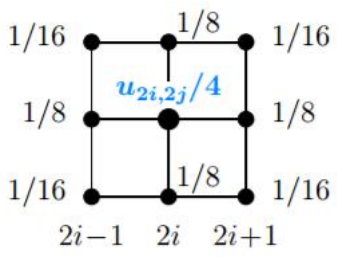


Figura 3: Full Weighted Restriction 8 Vecinos

Donde $h = \frac{1}{N-1}$ y N es la cantidad de nodos de la grilla.
Luego de reemplazar la ecuación nos queda de la forma:

$$\frac{u_{i-1,j} + u_{i,j-1} - 4u_{i,j} + u_{i+1,j} + u_{i,j+1}}{h^2} = f_{i,j} \tag{22}$$

7. Implementación

Para la implementación de los algoritmos se utilizó **Python** con el paquete **numba** para optimizar el código computacionalmente exigente.

El algoritmo de **Gauss Seidel** se implementó utilizando la discretización previamente descrita agregando el índice n de la iteración actual y $n+1$ de la iteración siguiente, por lo que la ecuación nos queda de la forma

$$u_{i,j}^{n+1} = \frac{-h^2 f_{i,j} + u_{i-1,j}^{n+1} + u_{i,j-1}^{n+1} + u_{i+1,j}^{n} + u_{i,j+1}^{n}}{4} \tag{23}$$

Suponiendo que recorremos la solución de arriba hacia abajo, de izquierda a derecha los valores de $u_{i-1,j}^{n+1}, u_{i,j-1}^{n+1}$ ya han sido calculados anteriormente, por lo que no necesitamos realizar operaciones más costosas para obtenerlos.

Para el caso de **V-Cycle** y **Full Multigrid** se implementaron las funciones de restricción *Injection* y *Full Weight* con 8 vecinos (Ver figura 3), una función de interpolación que ocupa los 8 vecinos a un punto en la malla fina.

8. Experimentos

Para comparar el comportamiento de estos algoritmos se realizaron dos experimentos

1. Ver la cantidad de iteraciones, relajaciones Gauss Seidel equivalentes, error y tiempo de los tres algoritmos variando la cantidad de nodos de la discretización del dominio, además se probaron ambas restricciones para ver si existia un cambio en el comportamiento del algoritmo.

2. Variar los parámetros ν_0, ν_1, ν_2 dejando fija la cantidad de nodos para ver el impacto de estos parámetros.

9. Resultados y Conclusiones

9.1. Experimento 1

Lo que se busca en este experimento es ver qué tan eficaces son los 3 algoritmos descritos anteriormente. En este experimento se corrieron los distintos algoritmos con $n = 65, 129, 257$.

Como era de esperarse la cantidad de iteraciones de cada algoritmo aumenta con la cantidad de nodos, en el caso de Gauss Seidel sólo esta cantidad crece exponencialmente a medida que aumenta la cantidad de nodos, para V-Cycle y FMG se ve que la cantidad de iteraciones aumenta casi de forma lineal utilizando Full Weighted y Injection como funciones de restricción como se puede ver en las figuras 4 y 5.

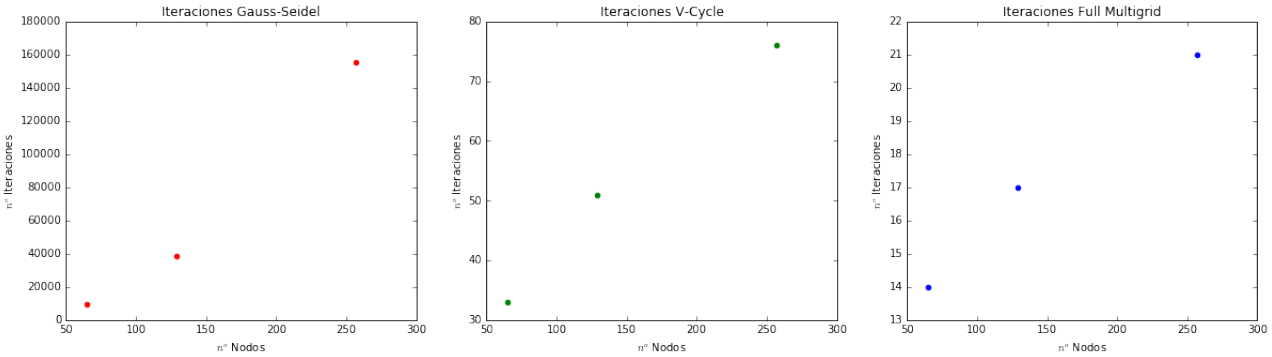


Figura 4: Número de iteraciones Full Weighted Restriction

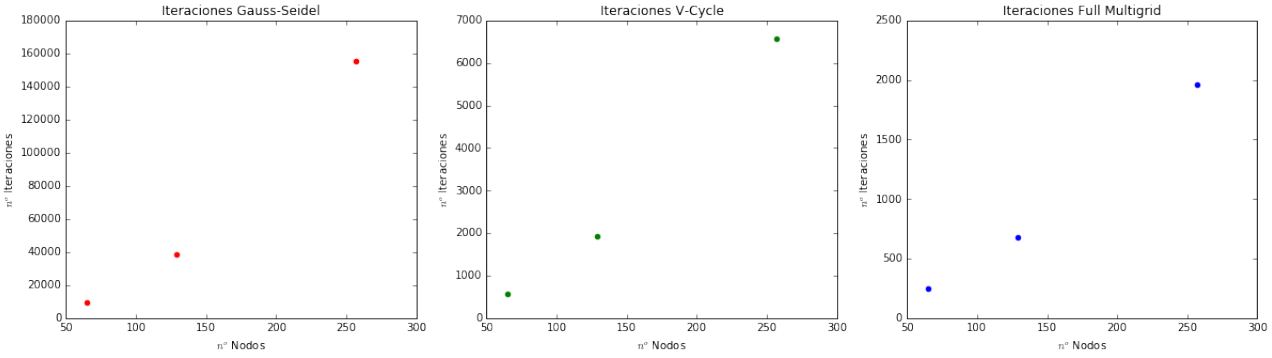


Figura 5: Número de iteraciones Injection Restriction

Pero realizar la comparacion de ciclos de cada algoritmo no es muy representativa, por lo que se calcularon en V-Cycle y FMG la cantidad equivalente de relajaciones Gauss Seidel en la malla fina que se realizaron en total, de lo que resultan las figuras 6 y 7. De estos resultados se aprecia que Gauss Seidel sigue teniendo una gran cantidad de ciclos en comparación a los otros algoritmos, siendo FMG el con menor cantidad de iteraciones equivalentes.

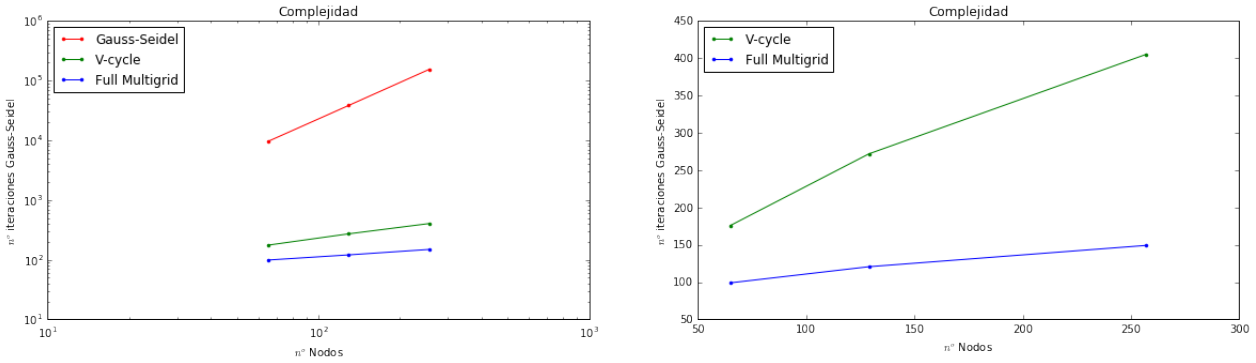


Figura 6: Número de iteraciones equivalentes Full Weighted Restriction

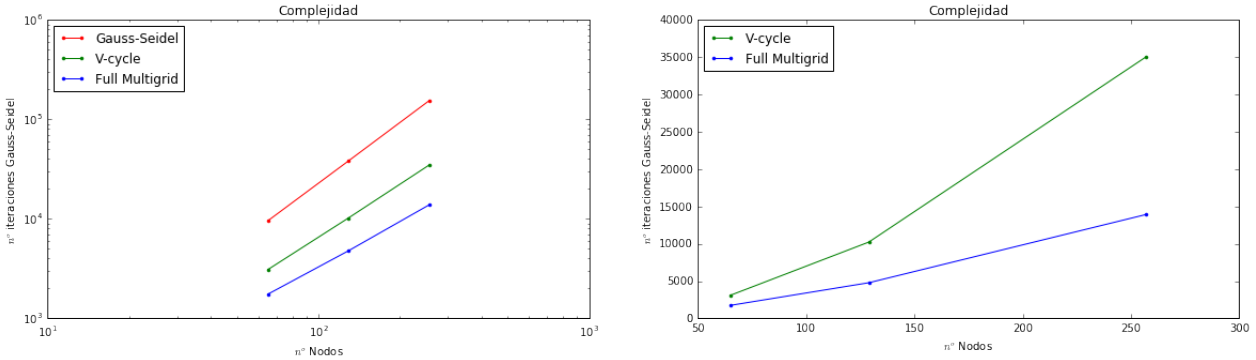


Figura 7: Número de iteraciones equivalentes Injection Restriction

Si dejamos de lado las iteraciones y nos centramos en el tiempo en escala Log-Log (Figuras 8 y 9). En este caso si podemos observar el comportamiento cuadrático de Gauss Seidel en comparación al lineal de V-Cycle y FMG, pero sólo en el caso de Full Weighted, para Injection la cantidad de tiempo se comienza a acercar. Teniendo en cuenta que el comportamiento lineal teórico de estos algoritmos no toma en cuenta el manejo de memoria al momento de cambiar la grilla podemos decir que el resultado práctico se acerca al teórico.

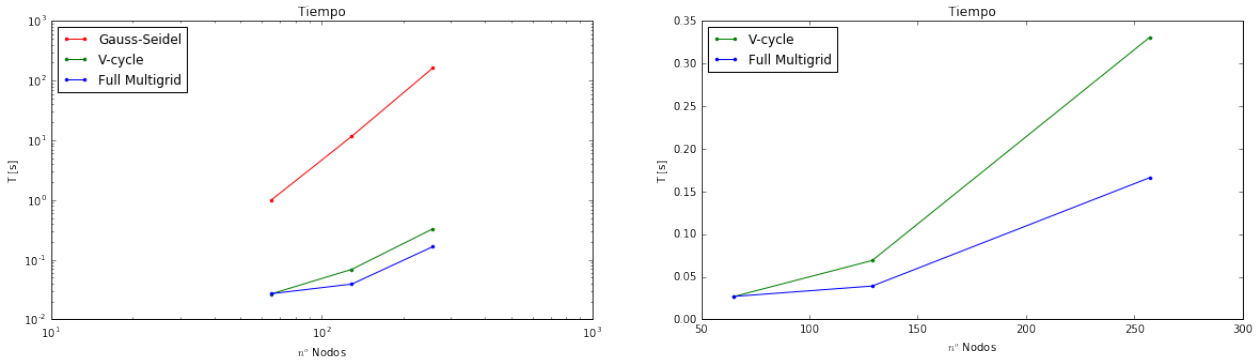


Figura 8: Tiempo con Full Weighted Restriction

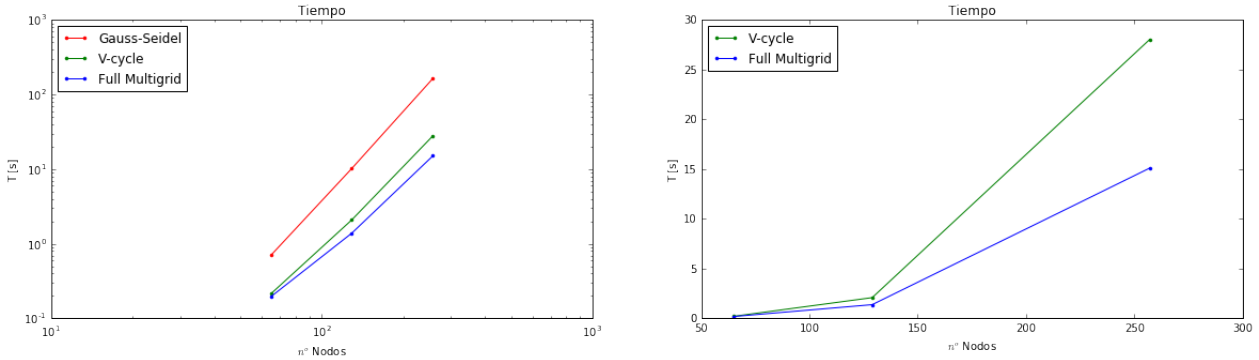


Figura 9: Tiempo con Injection Restriction

Como se obtuvieron buenos resultados para V-Cycle y FMG se procedió a analizar que tan buena era la solución obtenida, como se observa en las figuras 10 y 11, en el caso de Gauss Seidel y V-Cycle se encontraron soluciones comparables con un bajo error, el problema se encuentra en el FMG que se estanca en soluciones de no tan buena calidad, esto puede ser debido a que el FMG implementado no resuelve el mismo problema en todas las mallas, por lo que encuentra una solución distinta y para estos casos peor.

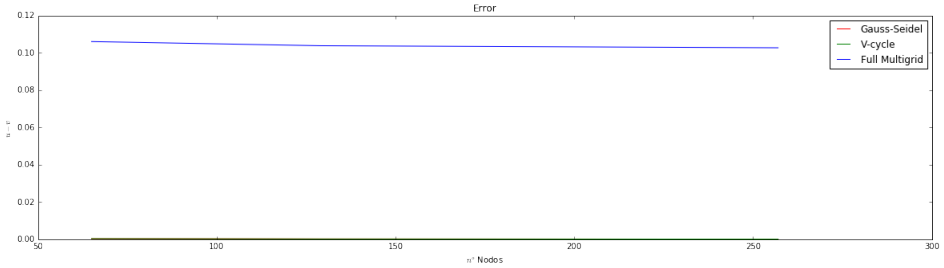


Figura 10: Error $e = |u - v|$ con Full Weighted Restriction

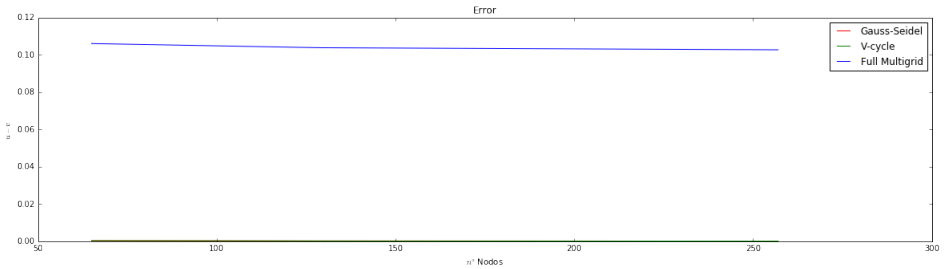


Figura 11: Error $e = |u - v|$ con Injection Restriction

Si observamos el residual de cada caso vemos que para Gauss Seidel (Figura 12) y V-Cycle (Figura 13) la mayor cantidad de error se encuentra en la zona central de la solución en cambio para FMG (Figura 14) está concentrado en un extremo de la solución.

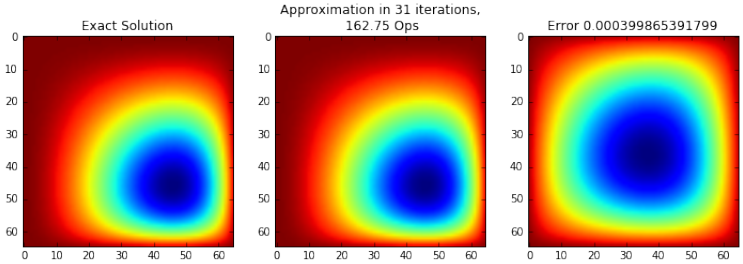


Figura 12: Error Gauss Seidel

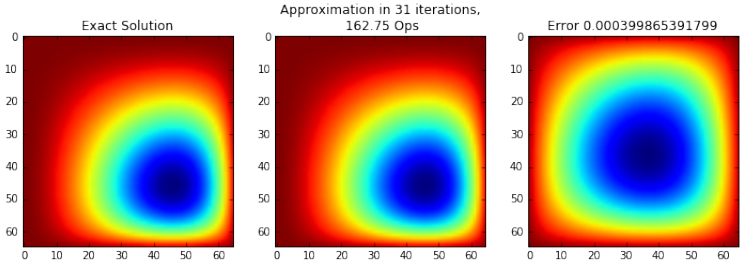


Figura 13: Error V-Cycle

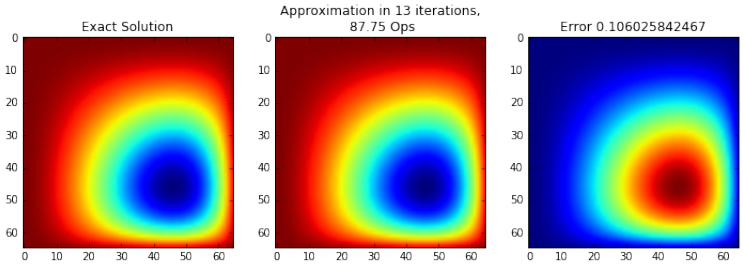
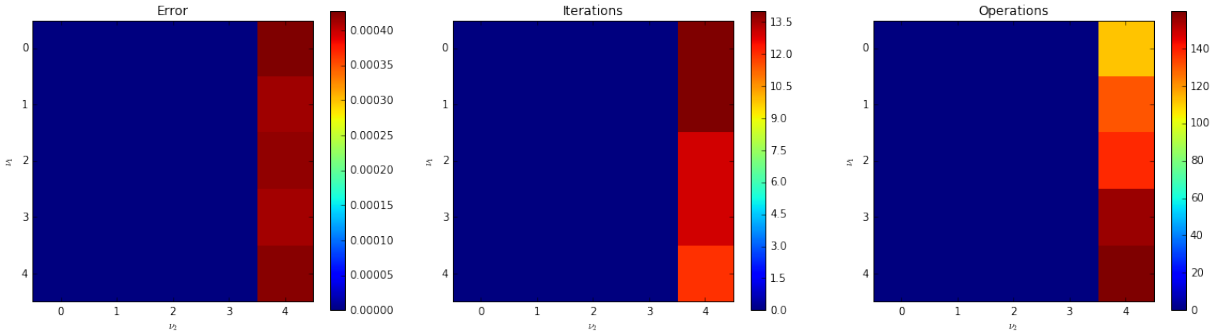


Figura 14: Error Full Multigrid

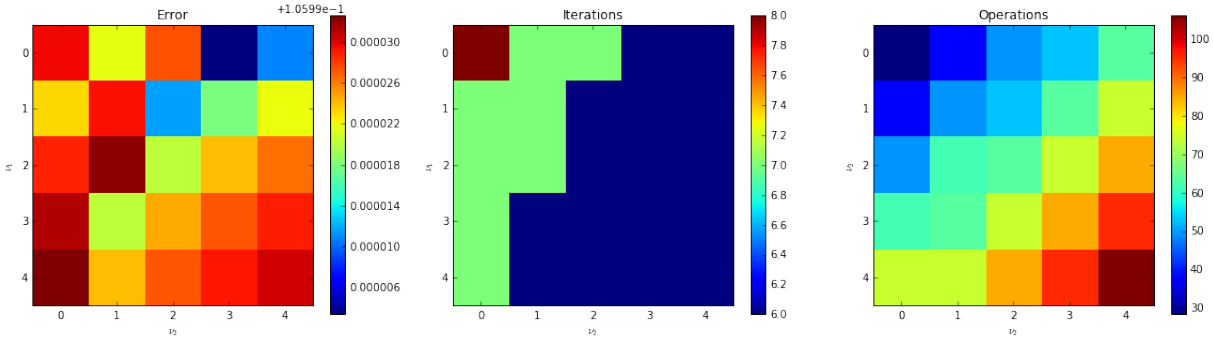
9.2. Experimento 2

En este experimento se buscaba ver cómo afectan los parámetros ν_0, ν_1, ν_2 a los distintos algoritmos, para esto se utilizaron los 3 algoritmos variando los parámetros.

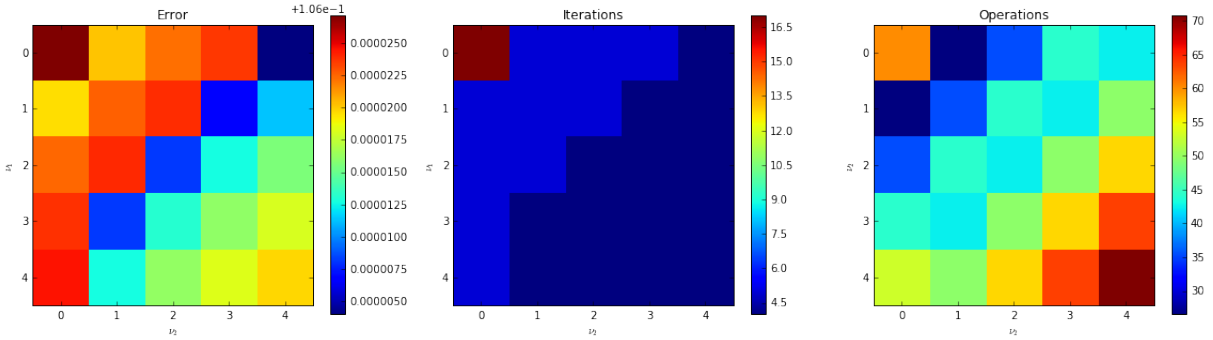
Como se ve en la figura 15, para V-Cycle independiente del parámetro ν_1 se obtuvo un buen resultado, pero cuando ν_2 es 5 se tiene que el error es mayor, como se observó en el experimento 1 la implementación de FMG realizada no es muy buena por lo tanto no se obtuvieron resultados concluyentes variando ν_0



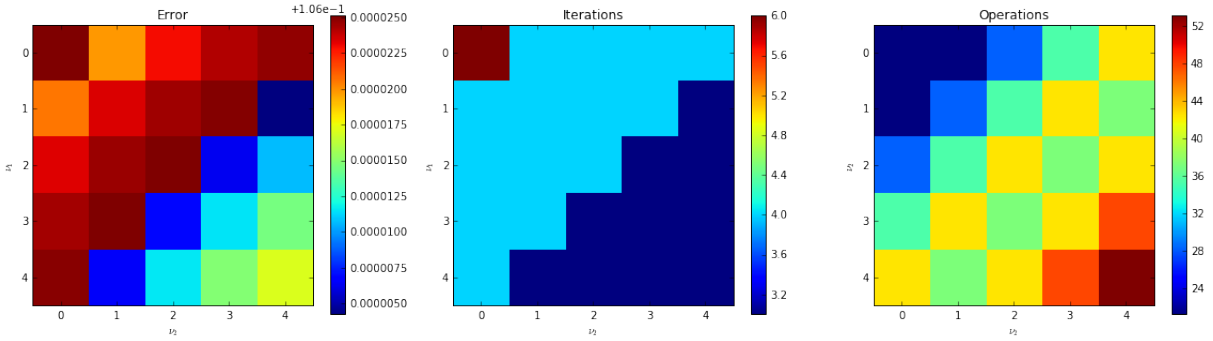
(a) V-Cycle



(b) FMG $\nu_0 = 1$



(c) FMG $\nu_0 = 2$



(d) FMG $\nu_0 = 3$

Figura 15: Experimento 2 variando ν_0, ν_1, ν_2

9.3. Conclusiones Generales

La utilización de algoritmos iterativos para resolver sistemas de ecuaciones lineales en especial Ecuaciones Diferenciales Parciales resulta útil ya que necesitan menor tiempo para poder obtener soluciones en comparación de algoritmos completos como descomposiciones PALU, Cholesky u otro. Puede ser que el error de este tipo de métodos sea mayor pero dependiendo de la aplicación esto puede no ser tan importante, ya que se necesitan buenas aproximaciónes más que resultados exactos.

Por su parte los algoritmos de Multi-Malla como V-Cycle o Full Multigrid ofrecen una convergencia mucho más rápida llegando a tener complejidad lineal al estar bien implementados. En el caso del algoritmo FMG implementado se observó que a pesar de tener una convergencia más rápida encuentra soluciones peores que las de V-Cycle o Gauss Seidel por lo que no es una buena implementación.

Referencias

[1] William L Briggs, Van Emden Henson, and Steve F McCormick. *A multigrid tutorial*. SIAM, 2000.

[2] Jong Chul Ye, Charles A Bouman, Kevin J Webb, and Rick P Millane. Nonlinear multigrid algorithms for bayesian optical diffusion tomography. *IEEE Transactions on Image Processing*, 10(6):909–922, 2001.