Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias Físico-Matemáticas

Desarrollo de método explícito para solución de la ecuación de Poisson en dominios de forma geométrica compleja

Tesis presentada al

Posgrado en Ciencias - Física Aplicada

como requisito parcial para la obtención del grado de

Maestro en Ciencias - Física Aplicada

por

Rogelio Paredes Jaramillo

asesorado por

Dr. Alexandre Grebennikov

Puebla Pue.
Junio 2015

Título: Desarrollo de método explícito para solución de la ecuación de Poisson en dominios de forma geométrica compleja

Estudiante: Rogelio Paredes Jaramillo

COMITÉ

Dra. Honorina Ruíz Estrada.
Presidente

Dr. Gerardo Francisco Torres del Castillo.
Secretario

Dr. Vladimir Alexandrov.
Vocal

Dr. José Fernando Rojas Rodríguez.
Vocal

Dr. Alexandre Grebennikov
Asesor

Índice general

Agradecimientos

Agradezco a cada uno
de los miembros del jurado
Dra. Honorina Ruíz Estrada, Dr.
Gerardo Francisco Torres del Castillo,
Dr. Alexandrov Vladimir y Dr. José Fernando Rojas Rodríguez quienes con sus comentarios y sugerencias me permitieron reforzar este trabajo. También quiero agradecer y reconocer el trabajo que realizan con los estudiantes de posgrado en la FCFM-BUAP a Dra. Marcela Maribel Méndez Otero y Dr. José Enrique Barradas Guevara. Agradezco a mi asesor Dr. Alexandre Grebennikov por su flexibilidad, comprensión y paciencia en cada una de nuestras sesiones de trabajo. Y por último agradezco a José Paredes Hernández, María Eustacia Jaramillo Flores† y a mis hermanos, quienes a lo largo de la vida y en las situaciones más difíciles me han mostrado una gran fortaleza para seguir adelante, apoyo incondicional
y me han motivado para concluir
esta etapa de mi vida.

Resumen

En el presente trabajo proponemos una serie de fórmulas para resolver la ecuación diferencial de Poisson y de Laplace en un dominio de solución con geometría compleja; validamos estas relaciones mediante experimentos numéricos, su fundamento teórico se ha obtenido recientemente en los trabajos de [1] [2]. Se presentan ejemplos y comparaciones entre la paquetería PDEToolbox y el método rayos generales que hemos programado en MATLAB.

Introducción

Para ayudarnos a comprender nuestro alrededor, a menudo describimos los fenómenos físicos por medio de relaciones matemáticas. Esta descripción es sólo una simplificación de algún modelo matemático, la cual no es del todo precisa; pero para fines prácticos puede proporcionarnos estimaciones para obtener una conclusión.

En general estas ecuaciones matemáticas que aparecen en los problemas de ciencias e ingenierías suelen ser ecuaciones diferenciales parciales (EDP) y para proporcionar información acerca del fenómeno que modelan han de ser resultas mediante métodos analíticos o numéricos. En [3] encontramos una clasificación de las ecuaciones diferenciales parciales lineales y a partir de éstas se sugiere una transformación o fórmula para hallar la solución analítica que depende del dominio de solución. Las soluciones analíticas asociadas pueden ser complicadas y en casos especiales necesitan mucho desarrollo analítico para obtenerse, el problema se incrementa cuando el dominio de solución es de geometría irregular.

El avance que han tenido las computadoras ha permitido el desarrollo de métodos numéricos para obtener una solución aproximada de las ecuaciones diferenciales parciales de manera mucho más rápida; además de que los métodos numéricos permiten abordar dominios de solución irregulares que muy difícilmente se resolverían en forma analítica. De los métodos numéricos mas empleados para dar solución a estas EDP están el método de diferencias finitas y el método del elemento finito. En [4] se presenta el método de diferencias finitas que generalmente se aplica a dominios de solución rectangulares y por tanto está limitado su uso a cierto tipo de problemas, en [9] se presenta el método de diferencias finitas para regiones curvas o semicirculares pero su aplicación sigue siendo limitada. En [5] y [6] se presenta el método de elementos finitos que ha tenido gran aceptación pues su formulación permite abordar cualquier dominio de solución, la limitante que tiene está en el sistema de ecuaciones lineales que se genera al resolver una EDP, además de que al requerir mayor precisión el sistema de ecuaciones se incrementa aumentado el tiempo de computo y el consumo de memoria.

En este trabajo se proponen una serie de fórmulas que constituyen el método de rayos generales (RG) para resolver la ecuación diferencial parcial de Poisson con condiciones de Dirichlet en un dominio de solución irregular, el método de rayos

generales permite resolver la ecuación diferencial parcial sin recurrir a un sistema de ecuaciones lineales. Los experimentos realizados muestran un menor tiempo de cómputo con respecto al método del elemento finito de PDEToolbox de Matlab; además de mostrar aproximadamente los mismos resultados numéricos.

Esta tesis está estructurada de la siguiente manera: en el capítulo 1 presentamos una breve revisión de las ecuaciones diferenciales parciales con la finalidad de definir el tipo de problema que vamos abordar en este trabajo.

En el capítulo 2 se presenta la formulación del método de rayos generales haciendo uso de las transformada directa e inversa de Radon. Posteriormente, en el capítulo 3 se presentan ejemplos numéricos con la finalidad de hacer comparaciones entre las soluciones obtenidas con PDEToolbox de Matlab y el método de RG implementado en Matlab, dichas comparaciones sólo son estimaciones gráficas ya que no hay una correspondencia precisa entre los nodos que discretizan el dominio de solución por elemento finito y la solución mediante rayos generales.

Finalmente procedemos a dar nuestras conclusiones en base a las simulaciones realizadas a lo largo del trabajo y se discute de manera breve el motivo por el cual dicha estimación es sólo gráfica y se plantean algunas cuestiones a resolver en próximos trabajos.

Concluimos con un apéndice con los respectivos scripts realizados en Matlab para la soluciones de las EDP presentadas en este trabajo.

Capítulo 1

EDPs y la ecuación de Poisson

Las ecuaciones diferenciales parciales (EDP) surgen al modelar fenómenos físicos generalmente en las áreas de ciencias e ingeniería, como ejemplos podemos mencionar la ecuación de conducción de calor, la ecuación de difusión y la ecuación de onda.

Una ecuación diferencial parcial es una expresión matemática que involucra una función incógnita de dos o más variables independientes y determina el comportamiento de la función en términos de estas variables independientes y sus derivadas parciales. El orden asignado a una EDP es el orden de la derivada parcial más grande. En el sistema cartesiano la forma más general de una EDP lineal de orden dos, con dos variables independientes tiene la forma:

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu = G \tag{1.1}$$

Como casos especiales, si tomamos B, C, D y F igual a cero y $A = 1$, $C = 1$, obtenemos la ecuación de Poisson; ecuación que se encuentra a menudo en la transferencia de calor, mecánica de fluidos, elasticidad y electrostática.

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = G(x, y) \tag{1.2}$$

otro caso especial ocurre cuando tomando $G = 0$ en la ecuación (1.2) obtenemos la ecuación de Laplace

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{1.3}$$

A la solución de la ecuación (1.3) se les suele llamar función armónica.
Frecuentemente las ecuaciones (1.2) y (1.3) son reescritas en otros sistemas de coordenadas que depende de la simetría del problema a resolver.

1.1. Condiciones iniciales y de frontera.

Una EDP tiene varias soluciones a menos que un conjunto de condiciones auxiliares sean impuestas para determinar la solución del problema particular que se resuelve. Tales condiciones están definidas por las condiciones físicas del problema que modelan y se clasifican de dos maneras, condiciones iniciales y de frontera.

Una condición inicial define el estado físico del problema que modelamos en un momento particular t_0

$$u(x, y, t_0) = f(x, y) \tag{1.4}$$

Las tres condiciones en la frontera más conocidas son:

Condición de Dirichlet. Para esta condición el valor de u se especifica en la frontera.

$$u(x, y) = f(x, y)$$

Condición de Neumann. La derivada normal $\dfrac{\partial u}{\partial n}$ se especificada en la frontera

$$\frac{\partial u}{\partial n} = f(x, y)$$

Condición de Robin. Es una combinación lineal de u y su derivada normal

$$a\frac{\partial u}{\partial n} + bu = f(x, y)$$

A menudo al problema dado por la EDP de Laplace (1.3) y las condiciones en la frontera de Dirichlet se le conoce como problema de Dirichlet.

Como mencionamos anteriormente, para estimar el comportamiento del fenómeno que una EDP modela, debemos dar paso a su solución mediante algún método analítico o numérico. En general la clasificación de una EDP determina la técnica a aplicar para su solución, que puede ser totalmente analítica o una aproximación numérica.

En [3] y [7] se puede hallar la solución analítica de las muchas ecuaciones diferenciales parciales que se encuentran en la física matemática incluyendo diferentes dominios de solución. Aún así, no siempre se puede obtener una solución y sí se obtiene generalmente se terminará evaluando en el ordenador hasta cierto número de términos.

1.2. Algunos métodos numéricos.

Cuando deseamos abordar un dominio de solución particular en el que la solución analítica es difícil de obtener, debemos recurrir a métodos numéricos como el método de diferencias finitas o el método del elemento finito; donde esté último es aplicable a dominios de solución mucho más generales.

1.2.1. Método de diferencias finitas y el método del elemento finito.

Para dar solución a una EDP bidimensional usando el método de diferencias finitas, debemos establecer en el dominio de solución una cuadrícula formada por dos conjuntos de líneas igualmente espaciadas $(\Delta x, \Delta y)$, una paralela al eje x y otra al eje y, reemplazar las derivadas parciales por sus aproximaciones en diferencias finitas y obtener las soluciones aproximadas en los puntos de intersección (nodos), resolviendo N ecuaciones algebraicas con N incógnitas. Al incrementar el número de nodos se incrementa la resolución y por tanto la precisión en la solución numérica, (aunque el número de nodos estará limitado al esquema numérico empleado y su correspondiente condición de estabilidad), como consecuencia se tendrá más tiempo de computo y se demandará más memoria.

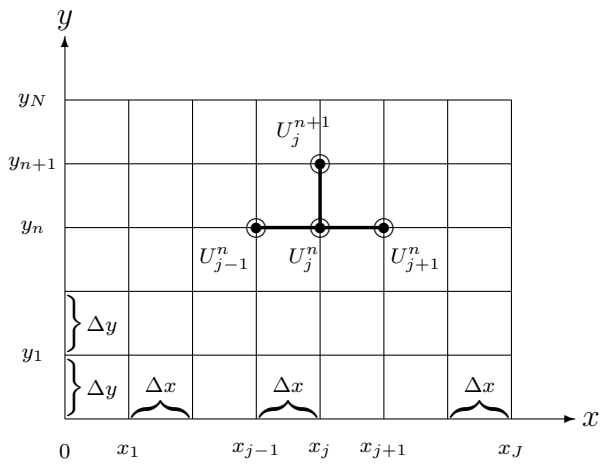


Figura 1.1: Discretización bidimensional del dominio de solución usando el método de diferencias finitas

Por otra parte el método del elemento finito permite abordar cualquier dominio de solución ya que su formulación descompone el dominio de solución mediante triángulos irregulares. Cada nodo representa el valor buscado de la solución numérica. Los pasos principales en el método del elemento finito se resumen en:

1. Discretizar el dominio de solución

2. Determinar las funciones de interpolación
3. Calcular los elementos matrices y vectores
4. Ensamblar las ecuaciones o el sistema global
5. Resolver el sistema de ecuaciones global.

Cada uno de los métodos anteriormente descritos tiene sus respectivas ventajas e inconvenientes. Si lo que deseamos es un cálculo para una dominio de solución rectangular sin requerir una gran precisión el método recomendado es el de diferencias finitas, método que por su formulación no demanda conocimientos avanzados de programación o implementación. Si lo que tenemos es una ecuación en un dominio de geometría compleja lo recomendado es el método del elemento finito, método que resulta ser más preciso pero demanda mayor conocimiento en su implementación y además de que al requerir mayor precisión incrementan los tiempos de cálculos.

1.3. Región estrellada simple

En este trabajo presentamos la solución numérica a la ecuación de Poisson (1.2) y de Laplace (1.3) usando el método de rayos generales, método que sólo se aplica a dominios que se llaman estrellados simples ó no estrellados simples y que a continuación definimos:

Región estrellada simple Llamaremos región estrellada simple a la región formada por una curva cerrada, suficientemente suave, dos veces diferenciable y desde la cual podemos hacer pasar un rayo desde un punto interior centrado en el origen hacia la frontera y no existen más de una intersección. Ver figura (1.2).

Región no estrellada simple Una región que no cumple las propiedades de una región estrellada simple es llamada región no estrellada simple. En la figura (1.3) el radio que parte del origen tiene dos intersecciones y por lo tanto es una región no estrellada simple.

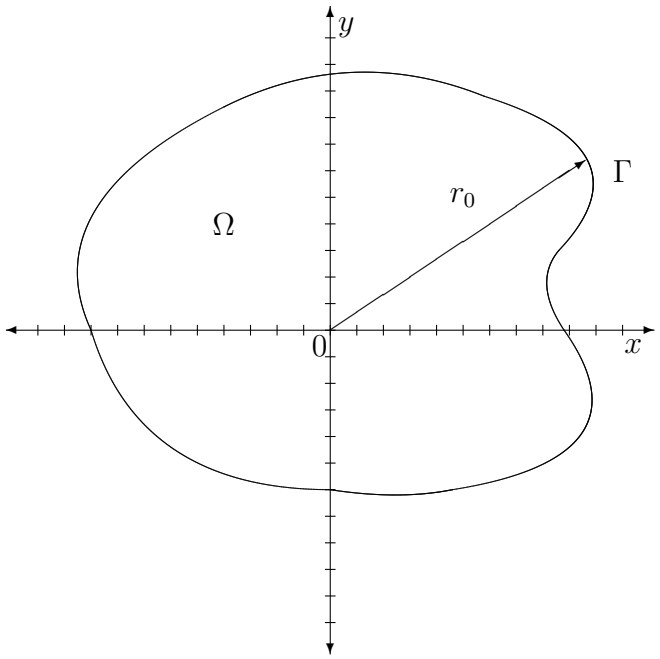


Figura 1.2: **Región estrellada simple.** Ω con frontera Γ

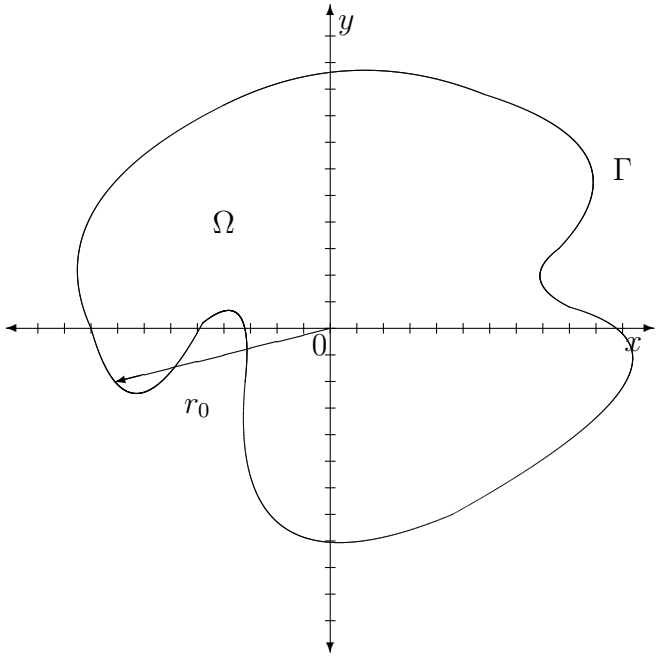


Figura 1.3: **Región NO estrellada simple.** Ω con frontera Γ

Capítulo 2

El método de rayos generales

En este capítulo describimos la transformada directa e inversa de Radon, el método de rayos generales (RG) y su aplicación en la solución de la ecuación de Poisson (1.2) y de Laplace (1.3) en un domino con geometría compleja.

2.1. Transformada de Radon

En 1917 Johann Radon propuso cómo construir una función de dos variables a partir de todas las integrales de línea sobre el plano. No fue hasta 1960 que la transformada de Radon tomó un papel importante al usarse en la tomografía para reconstruir las secciones transversales del interior de un objeto sin tener que cortarlo o dañarlo.

La transformada de Radon para una función de dos variables $f(x, y)$ se define en [10] y en [11] como la integral de línea de $f(x, y)$ para todas las líneas l definidas por los parámetros ϕ y p como se ilustra en la figura 2.1. En términos de integrales de línea sobre l tenemos:

$$\breve{f}\,(p, \phi) = \Re f = \int_{-\infty}^{\infty} f(\mathbf{r}) dl \tag{2.1}$$

donde $\mathbf{r} = (x, y)$ es un vector de posición. Una manera equivalente de escribir la transformada de Radon usando la función delta y considerando que la ecuación de la recta l esta dada por $x \cos\phi + y \operatorname{sen}\phi - p = 0$:

$$\breve{f}\,(p, \phi) = \Re f = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f\,(x, y)\, \delta\,(p - x\cos\phi - y \operatorname{sen}\phi)\, dxdy \tag{2.2}$$

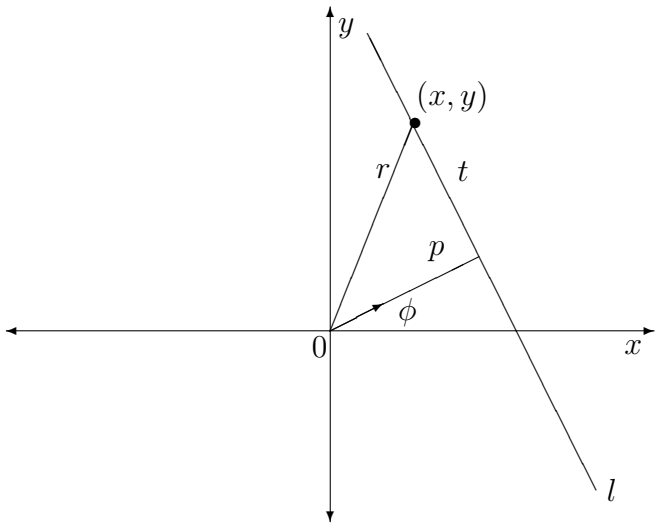


Figura 2.1: Sistema de coordenadas cartesiano en el que la ecuación de la recta esta dada por $p = x\cos\phi + y\text{sen}\,\phi$

Si consideramos un nuevo sistema de coordenadas como el de la figura 2.2 pero ahora rotando un ángulo ϕ, de tal manera que el eje p sea perpendicular a la línea dada por $x\,\cos\phi + y\,\text{sen}\,\phi - p = 0$ tenemos otra forma de escribir la transformada de Radon:

$$\breve{f}\,(p,\phi) = \Re f = \int_{-\infty}^{\infty} f(p\cos\phi - t\text{sen}\,\phi, p\text{sen}\,\phi + t\cos\phi)dt \tag{2.3}$$

La ecuación anterior tiene la siguiente interpretación: si $f_\phi\,(p,t) = f\,(p\cos\phi - t\text{sen}\,\phi, p\text{sen}\,\phi + t\cos\phi)$ es la representación de la función $f\,(x,y)$ con respecto al sistema de coordenadas rotado por un ángulo ϕ, la correspondiente transformada de Radon es la integral de $f_\phi\,(p,t)$ con respecto a t para el ángulo fijo ϕ.

$$\breve{f}_\phi(p) = \int_{-\infty}^{\infty} f_\phi(p,t)dt \tag{2.4}$$

En [10] se presentan las propiedades de linealidad, similaridad, simetría, traslación, diferenciación y convolución para la transformada de Radon.

2.2. Método de rayos generales

El método de rayos generales (RG) consiste en construir para cada EDP un conjunto de ecuaciones diferenciales ordinarias (EDO) a lo largo de rayos, estos rayos están representados por una recta l usando la parametrización de Radon, con

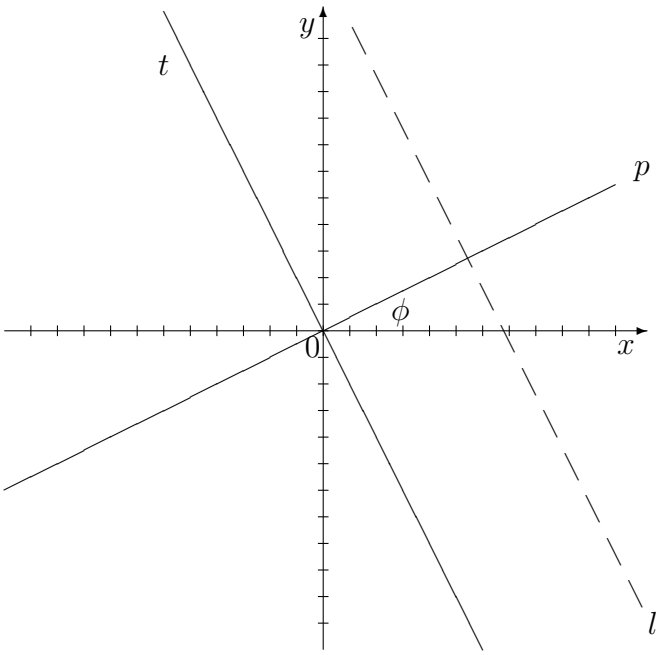


Figura 2.2: Sistema de coordenadas p - l que se a rotado un ángulo ϕ

parámetro t (fijado el ángulo ϕ); es decir $x = p\cos\phi - t\text{sen}\,\phi$, $y = p\text{sen}\,\phi + t\cos\phi$. Aquí $|p|$ es la longitud de la perpendicular que va desde el origen a la línea l y $\phi \in [0, \pi]$ es el ángulo entre el eje x y su perpendicular.

En este trabajo nos ocuparemos de la ecuación de Poisson con condiciones en la frontera de Dirichlet

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = G(x, y), \qquad x, y \in \Omega \tag{2.5}$$

$$u(x, y) = f(x, y), \qquad x, y \in \Gamma \tag{2.6}$$

suponemos que la función $u(x, y)$ tiene derivadas continuas en las variables x y y dentro del dominio de solución Ω y que este dominio esta limitado por una curva continua Γ. Aquí $G(x, y) \in \Omega$ y $f(x, y) \in \Gamma$ son dos funciones dadas.

Para aplicar el método de RG al problema 2.5-2.6 debemos seguir los siguientes pasos:

1. Reducir el problema de valores en la frontera a uno homogéneo.

2. Describir la distribución de la función $u(x, y)$ a lo largo de rayos generales (líneas rectas) por su transformada de Radon $u_\phi(p)$.

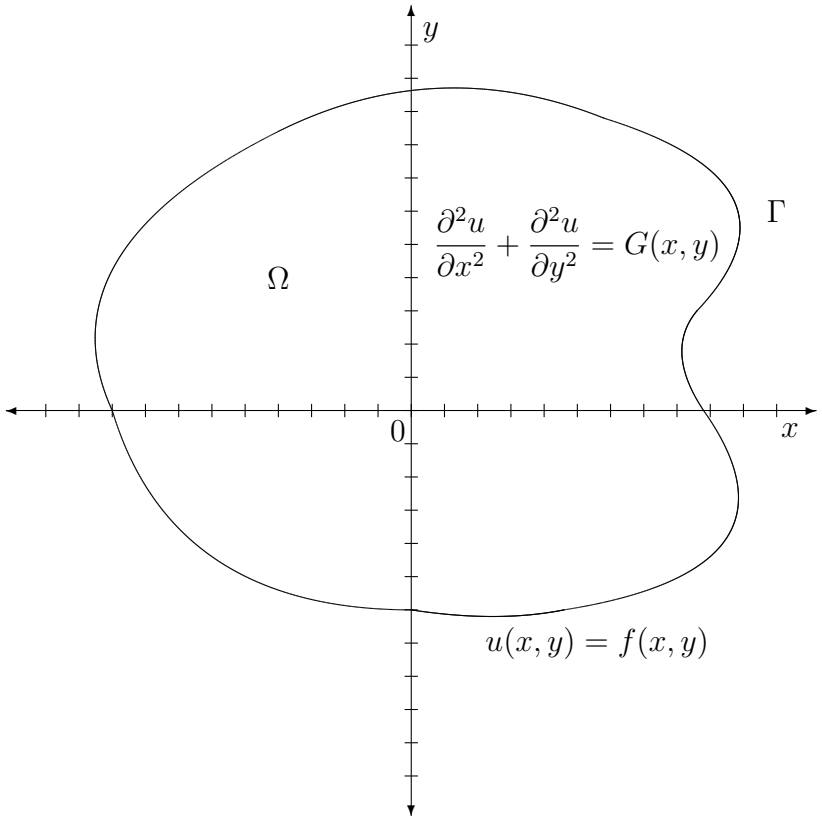


Figura 2.3: Dominio de solución

3. Construir una familia de EDO en la variable p con respecto a la función $u_\phi(p)$.
4. Dar solución a las EDO con condiciones en la frontera homogéneas.
5. Calcular la transformada inversa de Radon de las soluciones obtenidas para cada una de las EDO.
6. Reinversión de las condiciones iniciales en la frontera.

Comenzamos desarrollando el método suponiendo que la frontera Γ puede ser escrita en coordenadas polares (r, ϕ) por una función positiva univaluada, que denotamos por $r_0(\phi)$, $\phi \in [0, 2\pi]$ Ver figura 2.4.

Para la región estrellada simple Ω, cuyo centro está en el origen, es posible escribir la condición en la frontera como:

$$\overline{f}(\phi) = f(r_0(\phi)\cos\phi, r_0(\phi)\text{sen}\,\phi) = f(x_0, y_0) \tag{2.7}$$

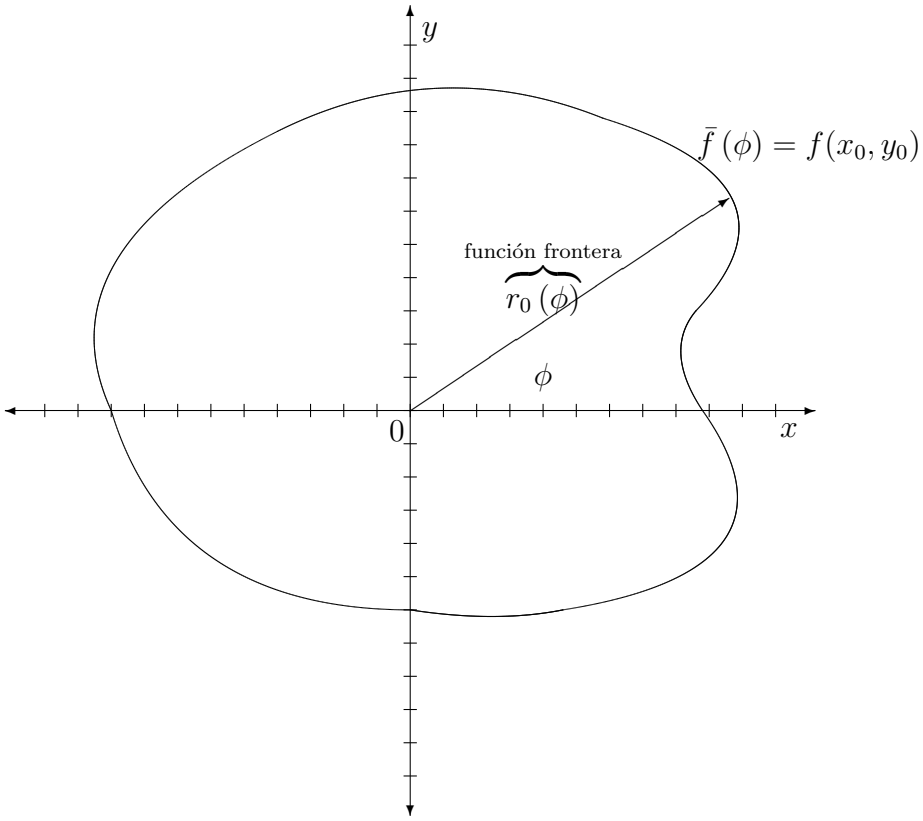


Figura 2.4: La función que describe nuestra condición en la frontera es $\overline{f}(\phi)$

Suponiendo que la función frontera $r_0(\phi)$ y la condición en la frontera $\overline{f}(\phi)$ tienen segunda derivada, proponemos las siguientes funciones con la finalidad de reducir el problema (2.5 - 2.6) a un problema de Dirichlet con condición en la frontera homogéneo.

$$f_0(\phi) = \frac{\overline{f}(\phi)}{r_0^2(\phi)} \qquad (x,y) \in \Omega \qquad r_0(\phi) \neq 0 \tag{2.8}$$

$$G_0(x,y) = G(x,y) - 4f_0(\phi) - f_0^{''}(\phi) \qquad (x,y) \in \Omega \tag{2.9}$$

$$u_0(x,y) = u(x,y) - r^2 f_0(\phi) \qquad (x,y) \in \Gamma \tag{2.10}$$

Mediante las relaciones propuestas (2.8-2.10) la ecuación de Poisson con condiciones de Dirichlet se reduce a:

$$\frac{\partial^2 u_0}{\partial x^2} + \frac{\partial^2 u_0}{\partial y^2} = G_0(x,y), \qquad (x,y) \in \Omega \tag{2.11}$$

$$u_0(x,y) = 0, \qquad (x,y) \in \Gamma \tag{2.12}$$

Una vez reducido el problema (2.5-2.6) al equivalente (2.11-2.12) mediante las transformadas propuestas, suponemos que las funciones y sus derivadas están extendidas en todo el dominio de solución y solo son cero fuera de éste, aplicamos la transformada directa de Radon (2.1) para describir la función $G_0(x, y)$ a lo largo de líneas.

Tomando la transformada directa de Radon en cada uno de los miembros de la ecuación (2.11) podemos obtener (usando la fórmula (2) de la página 3 de [12]) la familia de EDO con respecto a la variable p:

$$\Re\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)=\frac{d^2u_\phi(p)}{dp^2}=\Re G_0(x,y)=\breve{G}_0(p,\phi), \qquad \in \widehat{\Omega} \tag{2.13}$$

donde $\widehat{\Omega}$ es el nuevo dominio de solución por el cambio de parámetros p, ϕ. En donde, $\phi \in [0, \pi]$, el módulo del parámetro p es igual al radio en coordenadas polares y cambia en los límites determinados por la curva frontera Γ. En el caso que estamos considerando, para cualquier valor fijo de ϕ, el parámetro p está entre los limites $-r_0(\phi-\pi) < p < r_0(\phi)$.

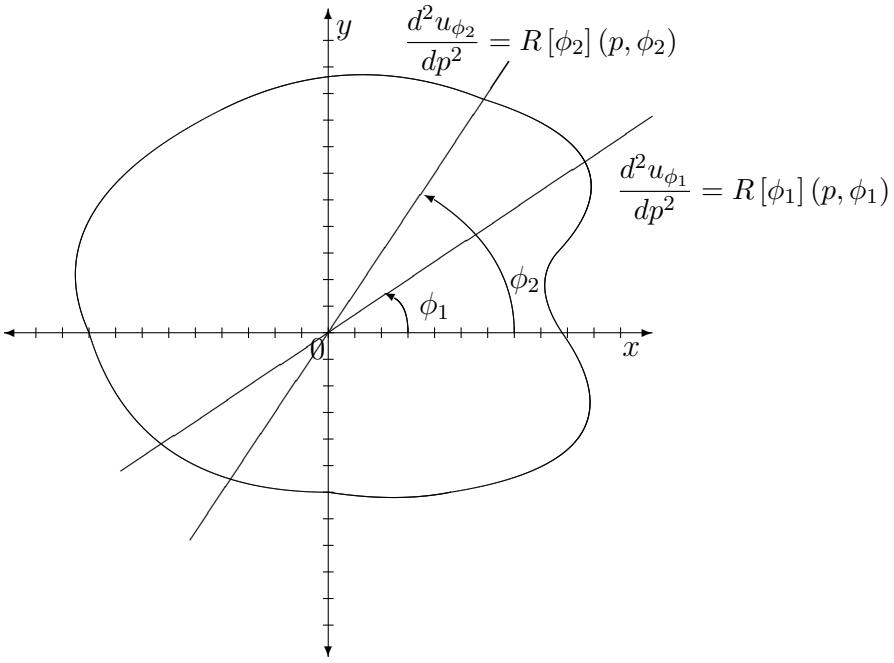


Figura 2.5: Sistema de ecuaciones en función del ángulo ϕ

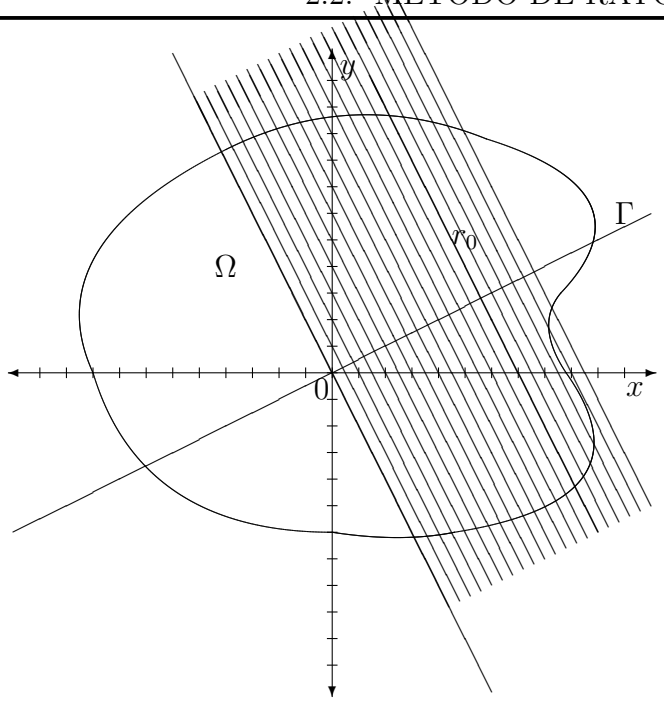


Figura 2.6: Cada integral de línea representa una EDO

La condición en la frontera (2.12) no puede ser modificada por la transformada directa de Radon para las correspondientes condiciones en la frontera de cada ecuación de la familia (2.13). Por lo tanto se propone el uso de las siguientes condiciones en la frontera para cada valor fijo de $\phi \in [0, \pi]$:

$$u_\phi(-r_0(\phi - \pi)) = 0; \quad u_\phi(r_0(\phi)) = 0 \tag{2.14}$$

Hasta aquí tenemos ya un conjunto de ecuaciones diferenciales ordinarias y sus respectivas condiciones iniciales, las cuales podemos resolver.

Designamos a $\widehat{u}_\phi(\phi)$ como la solución del problema (2.13)-(2.14) que puede estar unívocamente determinada como una función de la variable p para cada $\phi \in [0, \pi]$, $p \in (-r_0(\phi - \pi), r_0(\phi))$ y fuera de este intervalo $\widehat{u}_\phi(p) = 0$ para todo p continuo sobre ϕ

Una vez obtenida la solución del conjunto de ecuaciones diferenciales ordinarias, debemos revertir para obtener una aproximación de la solución del problema (2.11-2.12), esta solución está dada por la transformada inversa de la solución:

$$\overline{u}_0(x, y) = \Re^{-1}\widehat{u}_0(p) \qquad \in \widehat{\Omega} \tag{2.15}$$

la cual se obtiene sustituyendo la función definida por (2.15) en la parte izquierda de la ecuación (2.11) y haciendo uso del lema 2.1 de la página 3 de [12]; es decir:

$$\frac{\partial^2 \overline{u}_0}{\partial x^2} + \frac{\partial^2 \overline{u}_0}{\partial y^2} = \Re^{-1}[\frac{d^2 \widehat{u}_\phi(p)}{dp^2}] = \Re^{-1}[\Re G_0(p,\phi)] = G_0(x,y) \tag{2.16}$$

Por otra parte, para la condición en la frontera $\widehat{u}_\phi(p) = 0$, $(r_0(\phi-\pi), r_0(\phi)), \phi \in [0,\pi]$ hacemos uso del Teorema del soporte (página 10 del libro [12]) de donde se sigue que $\overline{u}_0(x,y) = 0$ para $(x,y) \notin \Omega$ y debido a la continuidad, se satisface la condición en la frontera (2.12).

Finalmente la solución aproximada $\overline{u}(x,y)$ del problema en la frontera (2.5)-(2.6) puede resumirse en la siguiente fórmula:

$$\overline{u}(x,y) = \Re^{-1}[(\widehat{G}_2(p,\phi)) - \frac{(p + r_0(\phi-\pi))}{(r_0(\phi) + r_0(\phi-\pi))}\widehat{G}_2(r_0(\phi),\phi)] + r^2 f_0(\phi) \tag{2.17}$$

donde $G_2(p,\phi)$ está definida por:

$$\widehat{G}_2(p,\phi) = \int_{-r_0(\phi-\pi)}^{p} \int_{-r_0(\phi-\pi)}^{p} \widehat{G}_0(p,\phi)dp \tag{2.18}$$

$$\widehat{\psi}(p,\phi) = \Re[G_0(x,y)]$$

La transformada directa e inversa de Radon dadas por las fórmulas (2.17) y (2.18) pueden realizarse numéricamente en Matlab usando los respectivos comandos `radon` e `iradon`. Para reconstruir la solución mediante `iradon` debemos señalar que Matlab hace uso del algoritmo Back-projection filtrado de manera conjunta con la transformada rápida discreta (FFRD) de Fourier [11] la cual proporciona rapidez en el método propuesto.

Capítulo 3

Ejemplos numéricos

En este capítulo presentamos ejemplo numéricos del método de rayos generales (RG) descrito en el capítulo anterior. Hacemos una comparación entre el método de RG que se ha programado en Matlab y el método del elemento finito que funciona mediante una interfaz gráfica llamada PDEToolbox de la paquetería de Matlab.

3.1. PDEToolBox

Antes de pasar a dar solución numérica a las ecuaciones diferenciales parciales de este capítulo por el método de rayos generales, damos una breve introducción de cómo se resuelven las EDP usando la interfaz gráfica de usuario (GUI), con la finalidad de comparar tanto gráficamente las soluciones de PDEToolbox con las del método de rayos generales y los tiempos de solución.

Una vez inciado MATLAB, en la ventana de comandos se carga la paquetería PDEToolbox escribiendo: `pdetool` y oprimiendo enter. Ya que ha aparecido la ventana de la figura (3.1), con la ayuda de la barra de herramientas y las opciones del menú seguimos los pasos que a continuación se indican de una manera muy general; En [8] y [13] se muestra detalladamente el uso de PDEToolbox para este tipo de geometrías y otras más sofisticadas.

Pasos generales en la solución de EDP mediante PDEToolbox:

- Definir el tamaño del plano y construir el dominio de solución con ayuda de las herramientas de dibujo: rectángulo, elipse y polígono 
- A partir del domino de solución se define su frontera con el botón de la barra de herramientas, con ayuda de la tecla *Shift* se selecciona cada uno de los segmentos de la frontera y se establece el tipo de condición desde el menú **Boundary** y la opción **Specify Boundary Conditions**.

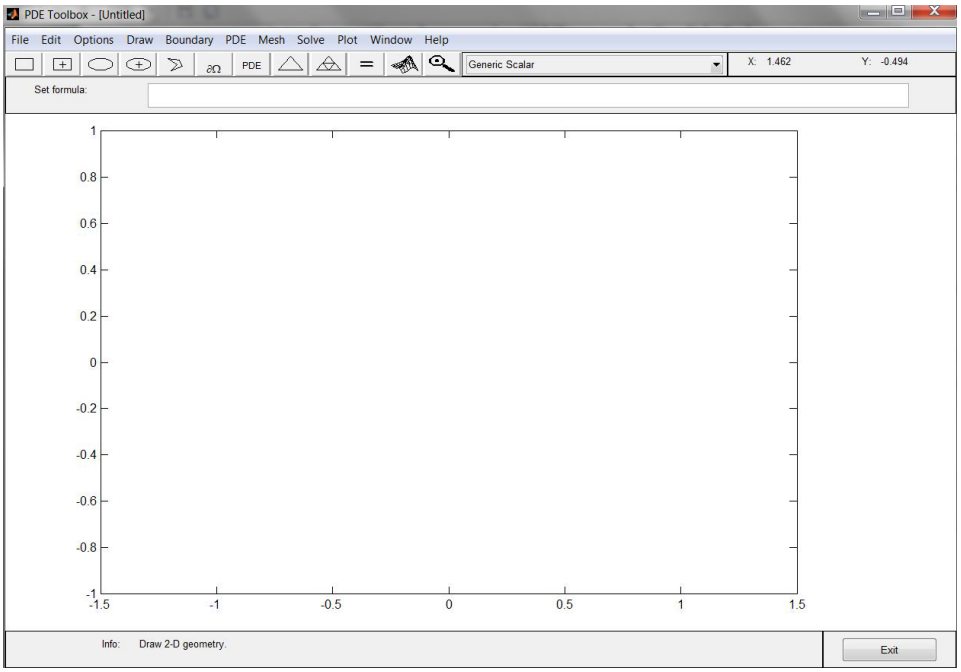


Figura 3.1: Interfaz PDEToolBox de MATLAB

- Establecidas las condiciones en la frontera se define el tipo de EDP que va a solucionarse (elíptica, parabólica o hiperbólica) pulsando el botón PDE; nos aparecerá una ventana para definir el tipo de ecuación y el valor para cada uno de sus coeficientes que se presentan.

- PDEToolbox funciona mediante el método del elemento finito, por tanto se establece la triangulación de manera automática al oprimir el botón △ o seleccionando del menú **Mesh** la opción **Initialize Mesh**. En caso de requerir una mayor precisión puede redefinirse la triangulación oprimiendo el botón ⧋; pero debe tenerse en cuenta que a mayor precisión se requiere un mayor tiempo de cómputo puesto que para cada nodo se genera una incógnita.

- Establecida la triangulación con el botón =, la combinación de teclas *Ctrl+e* o en el menú **Solve** se resuelve la EDP.

- Finalmente resuelta la ecuación de manera numérica, se deberá visualizar la gráfica de la solución usando la combinación de teclas *Ctrl+p* ó en el menú **plot** se deberá pedir la gráfica de la solución. Habilitando los parámetros Height (3-D plot), Show mesh y Plot in x-y grid.

La solución mediante PDEToolbox está condicionada a que el usuario tenga instalada dicha paquetería pues no está integrada en la instalación básica.

3.2. Ecuación de Poisson en región circular unitaria

Como primera comparación resolvemos la ecuación diferencial parcial de Poisson con condiciones en la frontera (3.1-3.2), cuyo dominio de solución es el circulo unitario mostrado en la figura (3.2)

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 1 \qquad x^2 + y^2 < 1 \tag{3.1}$$

$$u|_{x^2+y^2=1} = \frac{1}{4} \tag{3.2}$$

Para dar solución a esta ecuación hacemos uso de la simetría y la correspondiente transformación a coordenadas polares, por tanto reescribiendo tenemos:

$$\frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} = +1 \qquad 0 \leq r \leq 1,\ 0 \leq \theta \leq 2\pi \tag{3.3}$$

$$u|_{r=1} = \frac{1}{4} \tag{3.4}$$

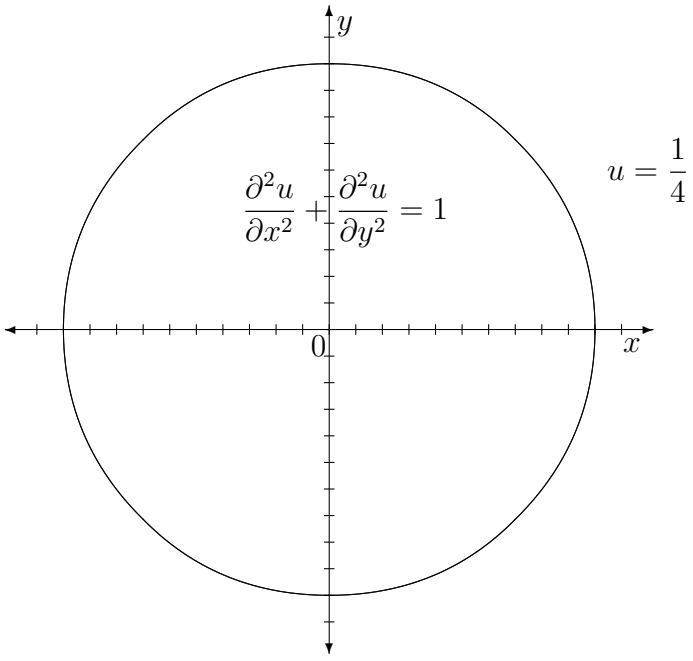


Figura 3.2: Circulo unitario para el problema de Poisson

La solución analítica al problema correspondiente (3.1)-(3.2) está dada por:

$$u = \frac{r^2}{4}$$

o en coordenadas cartesianas

$$u = \frac{x^2 + y^2}{4}$$

Ahora damos solución numérica al mismo problema (3.1)-(3.2) aplicando el método del elemento finito usando el paquete PDEToolbox (descrito en la sección anterior). Lo primero que hicimos es definir el dominio de solución y las condiciones en la frontera, posteriormente se tomó 549 para discretizar le dominio de solución. La triangulación del dominio de solución y el resultado numérico se muestra en las figuras 3.3-3.4.

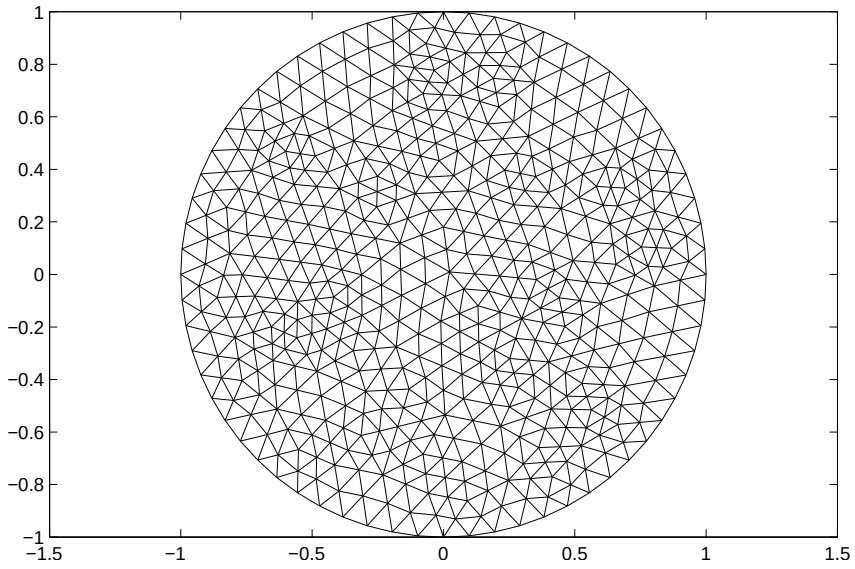


Figura 3.3: Discretización del dominio de solución con 549 nodos

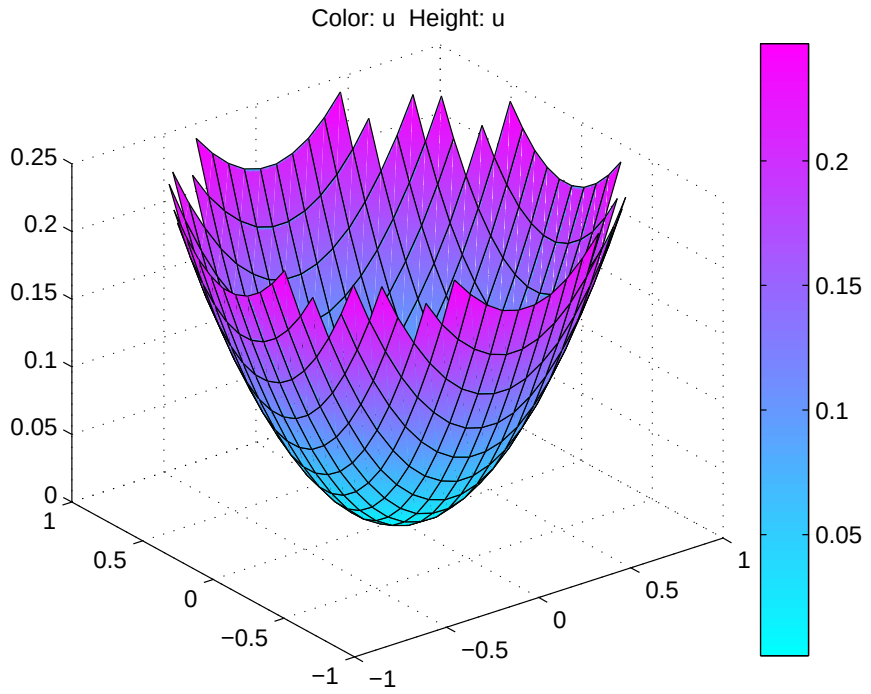


Figura 3.4: Solución numérica al problema (3.1)-(3.2) mediante PDEToolbox de Matlab

Finalmente damos solución al problema (3.1)-(3.2) pero ahora aplicando el método de rayos generales (RG), usando las fórmulas presentadas en el capítulo 2 en su forma discretizada. Para esto las formulas han sido programadas en scripts y se encuentran en el Apéndice, a continuación se describen de manera breve cada script:

poissoncirculoGRm.m Código principal que resuelve la ecuación de Poisson usando el método de rayos generales

f2.m En este script se define el segundo miembro de una EDP

fal.m Aquí se define la condición o condiciones en la frontera

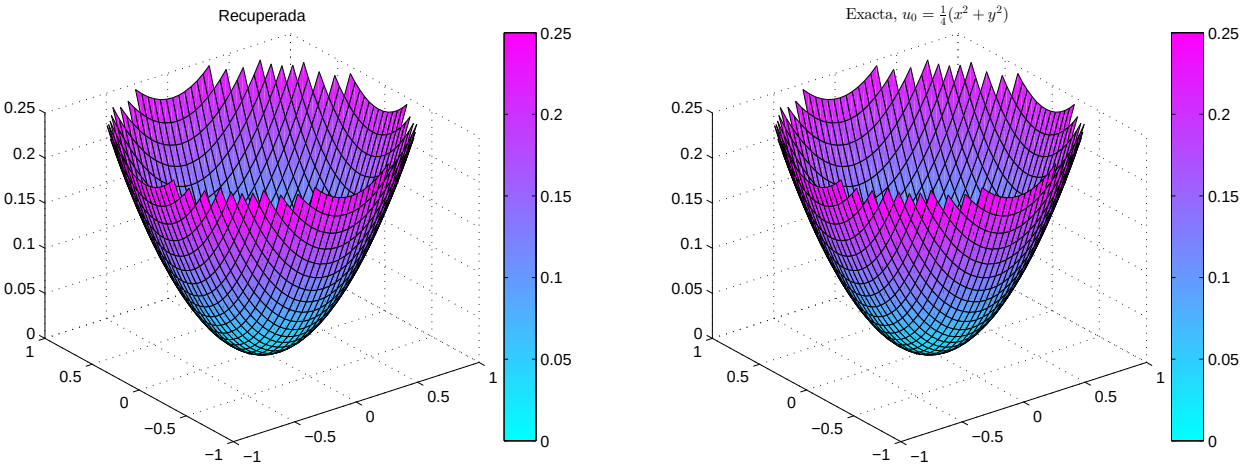
r02.m Aquí se define el dominio de solución

u0lapt1.m Aquí se escribe la solución analítica para compararla con la numérica

Para hacer las comparaciones entre el método de rayos generales y el método del elemento finito de PDEToolbox, tomamos el número de nodos arrojado por Matlab y a partir de este número calculamos el número de nodos equivalente que debemos usar en el método de RG, para esto usamos la fórmula propuesta en 3.5 que muestra correspondencia con las gráficas.

$$n = \sqrt{\frac{\text{nodos}}{0.6}} \tag{3.5}$$

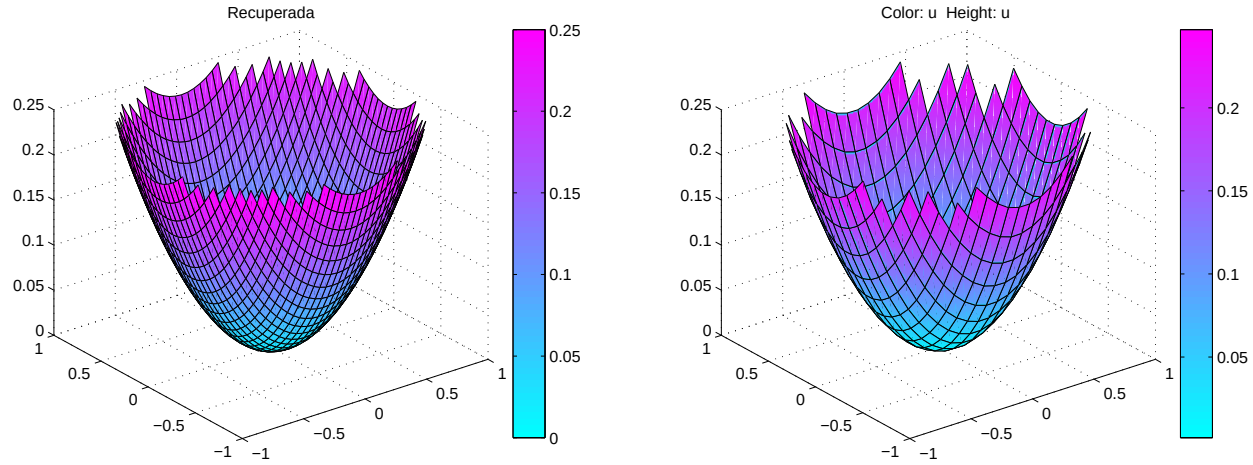
La figura 3.5 compara la solución por el método RG con la solución analítica, la Figura 3.6 compara la solución por el método de RG con el elemento finito.



(a) Solución mediante el método de rayos generales.

(b) Solución analítica.

Figura 3.5: Comparación gráfica entre método de rayos generales y solución analítica para el problema (3.1)-(3.2).



(a) Solución usando el método de rayos generales.

(b) Solución usando método del elemento finito.

Figura 3.6: Comparación gráfica entre método de rayos generales y el método del elemento finitas al problema (3.1)-(3.2)

Para concluir el ejemplo, comparamos los tiempos de cálculos entre método de RG y método del elemento finito. Para esto se corrieron ambos métodos en la misma computadora y se ejecutaron mediante línea de comandos. Como podemos ver considerando 26×26 nodos el método de rayos generales tiene una ventaja en cuanto a tiempo sobre el método del elemento finito de PDEToolbox que usa 549 nodos para el presente ejemplo.

```
Command Window
>> pdecirculopoisson549

tiempoPDETOOL =

    4.3634

>> poissoncirculoGRm(26,2)

tiempoGRm =

    0.4838

fx >>
```

Figura 3.7: Comparación de tiempos entre el método de rayos generales y el método de diferencias finitas en PDEToolbox

3.3. Ecuación de Poisson en región cruz

Como segundo ejemplo numérico presentamos el problema (3.1)-(3.2) pero ahora en un dominio de solución cruz:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 1 \qquad x^2 + y^2 < 0.45, \qquad x, y \in \Gamma$$

con condición de contorno en cada parte del dominio

$$\begin{aligned}
u|_{0.45^2+y^2} &= \frac{0.45^2 + y^2}{4} && -0.15 < y < 0.15 && && \Gamma_1 \\
u|_{0.15^2+y^2} &= \frac{0.15^2 + y^2}{4} && -0.45 < y < -0.15 && .015 < y < 0.45 && \Gamma_{11}, \Gamma_3 \\
u|_{0.15^2+y^2} &= \frac{0.15^2 + y^2}{4} && -0.45 < y < -0.15 && .015 < y < 0.45 && \Gamma_9, \Gamma_5 \\
u|_{-0.5^2+y^2} &= \frac{-0.5^2 + y^2}{4} && -0.15 < y < 0.15 && \Gamma_7 && \\
u|_{x^2+0.45^2} &= \frac{x^2 + 0.45^2}{4} && -0.15 < x < 0.15 && && \Gamma_4 \\
u|_{x^2+0.15^2} &= \frac{x^2 + 0.15^2}{4} && -0.45 < x < 0.15 && 0.15 < x < 0.45 && \Gamma_6, \Gamma_2 \\
u|_{x^2+(-0.15)^2} &= \frac{x^2 + (-.15)^2}{4} && -0.45 < x < 0.15 && 0.15 < x < 0.45 && \Gamma_8, \Gamma_{12} \\
u|_{x^2+(-0.45)^2} &= \frac{x^2 + (0.15)^2}{4} && -0.15 < x < 0.15 && \Gamma_{10} &&
\end{aligned}$$

La región cruz se muestra en la figura (3.8) con cada una de sus fronteras que conforma el dominio de solución. La diferencia con el ejemplo anterior radica en que para este caso no se tiene una solución analítica y por tanto nos limitaremos a las comparaciones numéricas entre el método de rayos generales y el método del elemento finito de PDEToolbox. Para dar solución a este problema enteramente numérico debemos ejecutar de manera conjunta los siguientes archivos pdecruzpoisson457.m y el archivo poissoncruzGrm.m con 28 nodos. En la figura 3.9 se muestra el dominio de solución discretizado con 457 nodos y la solución por elemento finito.

En la figura 3.10 se compara la solución numérica por el método de RG con la del método de diferencias finitas. Debemos resaltar que la gráfica por el método de RG tiene la misma forma; pero no es del todo igual a la obtenida por elemento finito. Finalmente mostramos los tiempos de comparación para la región cruz resaltando que el tiempo de calculo sigue siendo mayor usando elemento finito. Lo deseable sería comparar la solución numérica nodo a nodo pero debido a la discretización del elemento finito, tenemos triángulos irregulares los cuales no hacen posible comparar una discretización irregular con una cuadrícula.

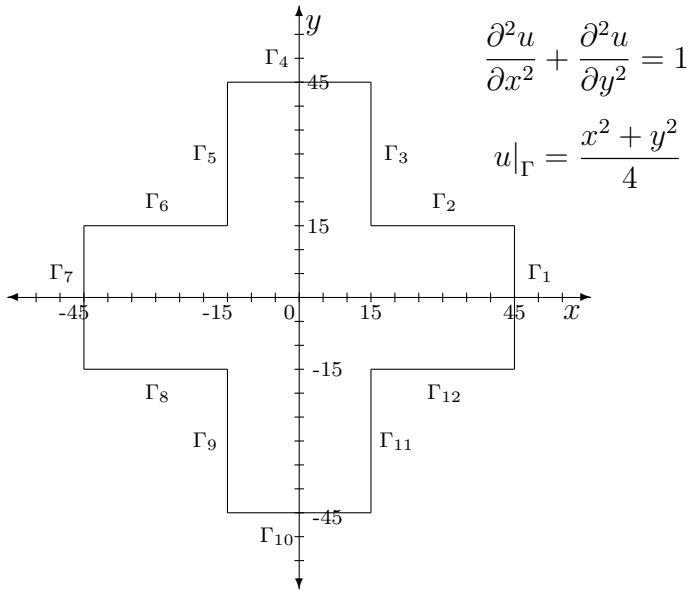
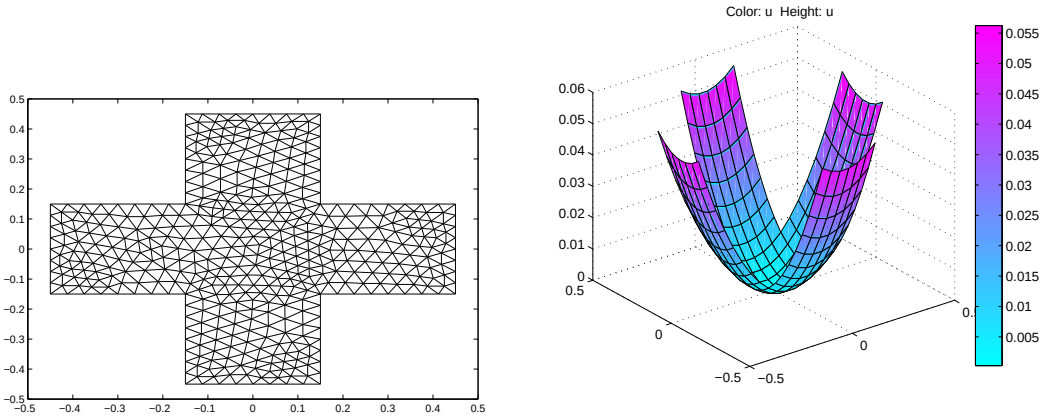


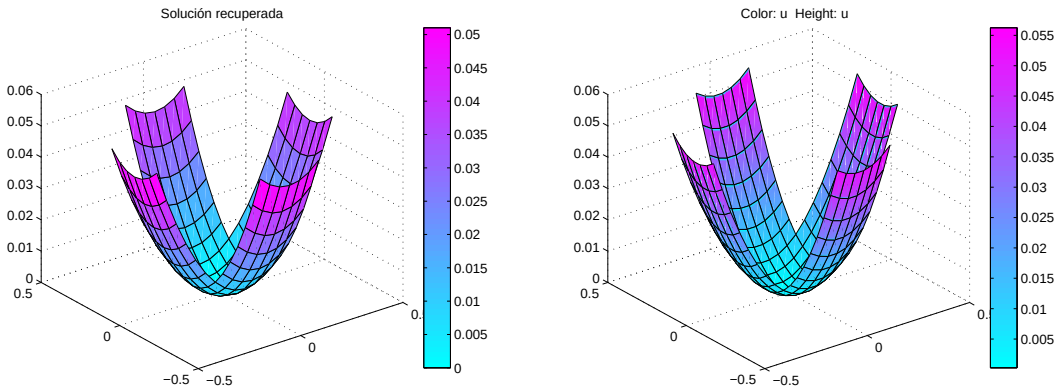
Figura 3.8: Dominio de solución para el problema de Poisson en región cruz.



(a) Discretización del domino de solución cruz

(b) Solución por el método del elemento finito

Figura 3.9: Solución numérica por método del elemento finito de PDEToolbox



(a) Solución mediante el método de rayos generales

(b) Solución por el método del elemento finito de PDEToolbox

Figura 3.10: Comparación gráfica entre método de rayos generales y el método del elemento finito para región cruz

```
Command Window
>> pdecruzpoisson457

tiempoPDETOOLBOX =

    2.4626

>> poissoncruzGrm(28,2)

tiempoGRm =

    0.8535

fx >>
```

Figura 3.11: Comparación de tiempos entre el método de rayos generales y el método del elemento finito en PDEToolbox

Conclusiones

Se ha presentado el método de rayos generales para dar solución numérica a la ecuación de Poisson en un dominio de solución de geometría "compleja" (circular y cruz), en cuanto a las relaciones cuantitativas no es posible hacer una comparación entre matrices de solución por ser algoritmos diferentes, en el método del elemento finito el dominio de solución se discretiza mediante triángulos irregulares y en el método de rayos generales se tiene como discretización puntos regulares en una cuadrícula.
De manera gráfica las soluciones por elemento finito y método de rayos generales son similares, se deja pendiente para otros trabajos la comparación numérica ya sea interpolando o por algún otro método que permita la mejor comparación. Al comparar los tiempos de solución entre el método de rayos generales con el método del elemento finito, se obtiene ventaja en el tiempo de cálculo usando el MRG pues en el primer ejemplo el tiempo de cálculo es aproximadamente nueve veces menor, en el segundo ejemplo solo se reduce el tiempo de cálculo en un tercio lo que se atribuye al dominio irregular de solución. El método de rayos generales tendrá una mejora en tiempo sobre el elemento finito principalmente porque en esta formulación no se resuelve un sistema de ecuaciones, sino una familia de ecuaciones diferenciales ordinarias.

A trabajos futuros sería bueno comparar el método de rayos generales con el método de diferencias finitas usando los mismos dominios de solución y el mismo número de nodos; así se facilitaría la comparación nodo a nodo y por tanto se podría estimar la precisión.
Finalmente resaltamos que este trabajo ha dado paso a ser presentado en extenso en el IV Congreso Internacional en Matemáticas Aplicadas realizado en el IPN [16] y publicado en Proceedings of the International Scientific-Technical Conference [17].

Capítulo 4

Apéndice

En esta sección presentamos los programas correspondientes al método de rayos generales usados en los experimentos numéricos, además anexamos el script correspondiente para el método del elemento finito de Matlab.

El programa principal para resolver la ecuación diferencial parcial 3.1-3.2, es `poissoncirculoGRM.m` que está compuesto por una serie de comandos y hacen uso de las funciones externas `f2.m` (que describe el segundo miembro de una ecuación diferencial parcial Poisson-Laplace), `fal.m` es donde se define la condición en la frontera, `r02.m` nos sirve para definir qué tipo de frontera vamos a trabajar, circulo o cruz. Finalmente `pdecirculopoisson549.m` es el archivo que activa la interfaz GUI de PDEToolbox para resolver la misma ecuación 3.1-3.2 pero esta vez usando el método de elementos finitos.

4.1. poissoncirculoGRM.m

```
function poissoncirculoGRm(n,nf);
% poissoncirculoGRM(n,nf) resuelve la ecuación de Poisson/Laplace
% en un circulo unitario o cruz usando el método de rayos generales.
% Los argumentos de entrada son:
% n= número de divisiones o rayos,
% nf es el numero de figuras a comparar al final i.e son las gráficas
% Se requiere las funciones f2.m, fal.m, r02.m , rfcCruzab.m y u0lap1.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic
% Con tic se inicia el contador de tiempo en t=0
h=2/(n-1);
% Se crea el espaciado de divisiones que habrá de manera vertical como
% horizontalmente considerando un cuadrado de lado 2 unidades
% centrado en el origen.
```

```
x=-1:h:1; y=x;
% Se crea un cuadrado cuyo centro es el origen y lados de magnitud 2
% unidades
hal=pi/(n-1); hfi=180/(n-1);
% Se crea el espaciado de divisiones para el ángulo de 180 grados i.e. el
% número de rayos que habrá
al0=0:hal:pi; fi=0:hfi:180;
% Se crean los vectores ángulo para trabajar las coordenadas polares
%**********
for i=1:n
  for j=1:n
   If(i,j)=f2(x(i),y(j));%f2 llama a las funciones externas fal.m y a r02.m
  end
end
%**********
% La función f2 toma cada uno de los nodos internos de la frontera definida
% en r02 lo que esta fuera de la frontera se anula según la
% función delta de dirac.
% fal es la condición en la frontera en coordenadas polares
% r02 define el dominio de solución, en este caso circulo unitario.
[rf,xp]=radon(If,fi); %Aquí es donde se aplica la transformada directa de Radon
nx=xp/max(abs(xp)); lxp=length(xp);
hp1=2/(lxp-1); %
rf2=0*rf; rf1=0*rf;
%**********
for j=1:lxp-1
    for i=1:j
        rf1(i+1,:)=rf1(i,:)+hp1*rf(i,:);
    end
    rf2(j+1,:)=rf2(j,:)+hp1*rf1(j,:);%
end
%*********
%*********
for i=1:n
  jr0=fix((-r02(al0(i)-pi)+1)/hp1)+1; %Se llama a r02.m que es el dominio sol.
  jr1=fix((r02(al0(i))+1)/hp1)+1;
     for j=1:lxp
        vp(j,i)=-(nx(j)-nx(jr0))/(nx(jr1)-nx(jr0))...
       *rf2(jr1,i)+rf2(j,i);
     end
     for j=1:jr0
        vp(j,i)=0;
     end
```

```
     for j=jr1:lxp
        vp(j,i)=0;
     end
end
%*********
urec=iradon(vp,fi,'nearest','Hann',1,lxp);%
surec=size(urec); lxp=surec(1);
hp1=2/(lxp-1);%
nx=-1:hp1:1;
%**********
for i=1:lxp
  for j=1:lxp
    r=sqrt(nx(i)*nx(i)+nx(j)*nx(j));
    al=atan2(nx(i),nx(j));
    u0(i,j)=u0lapt1(nx(i),nx(j)); %Se llama a la función u01lapt1.m
    urec0(i,j)=urec(i,j)+r^2*fal(al)/(r02(al))^2;
    ur00(i,j)=u0(i,j)-r^2*fal(al)/(r02(al))^2;
    d(i,j)=abs(u0(i,j)-urec0(i,j));
      if r>=r02(al)-hp1
         urec0(i,j)=r^2*fal(al)/(r02(al))^2;
      end
      if r>r02(al) % si algo esta fuera del dominio de solución, se anula
        d(i,j)=0;
        urec(i,j)=NaN;
        u0(i,j)=NaN;
        ur00(i,j)=NaN;
        urec0(i,j)=NaN;
     end
  end
end
%**********
% Se construyen las gráficas para compararlas
figure(nf) colormap(cool) surf(nx,nx,u0) axis([-1 1 -1 1 0 0.25])
colorbar('vert') title('Exacta,
${u_{0}=\frac{1}{4}(x^2+y^2)}$','interpreter','latex')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(nf+1) surf(nx,nx,urec0) axis([-1 1 -1 1 0 0.25])
colormap(cool) colorbar('vert') title('Recuperada') tiempoGRm=toc
```

4.2. f2.m

```
function rr=f2(x,y);
% f2(x,y) es la función del segundo miembro de una EDP
% Los argumentos de entrada son los nodos x y y
r=sqrt(x^2+y^2)+eps;
% Definimos el circulo unitario
al=atan2(x,y);
% Obtenemos el ángulo a1 para trabajar en coordenadas polares
n=121; %<-
h=pi/(n-1);
%%% Calculo aproximado de la derivada
rfal=fal(al)/r02(al)^2; % Aquí se llama a la tercera función externa r02
rfalmh=fal(al-h)/r02(al-h)^2; rfalph=fal(al+h)/r02(al+h)^2;
falp2=(rfalph-2*rfal+rfalmh)/h/h;
rr=1-4*rfal-falp2;%Si es Laplace el primer sumando es cero i.e el 1 se quita
%%*******************************
rr=0; if r>= r02(al)
    rr=0;
end
```

4.3. fal.m

```
function r1=fal(al);
% fal(al) es la condición en la frontera en coordenadas polares
% los parámetros de entrada son:
% a1 que es el ángulo a trabajar definido en f2.m
r1=r02(al)*r02(al)*(sin(al)*sin(al)+cos(al)*cos(al))/4;
% para circulo unitario u= (x*x+y*y)/4 en coordenadas polares es u=1/4:
% Aquí se escribe la condición en la frontera en coordenadas polares
```

4.4. r02.m

```
function rr=r02(al);
% r02(a1) define cual es el domino de solución en el MRG puede ser un
% circulo unitario o la región cruz
% Los parámetros de entrada son:
% a1=constante, en este caso el radio es r=1
rr=1;%Descomentar si trabajamos con circulo unitario
%rr=rfcCruzab(0.15,0.45,al); Descomentar si trabajamos con cruz
```

4.5. rfcCruab.m

```
function rr=rfcCruzab(a,b,al);

if al<-pi
   al=al+2*pi;
end if al>=pi
   al=al-2*pi;
end

al1=-pi/2; al2=-atan(b/a); al3=-pi/4; al4=-atan(a/b); al5=-al4;
al6=pi/4; al7=atan(b/a);

 if (al>=-pi-al2)&(al<=al2)
    rr=b*sqrt(1+1./(tan(al).*tan(al)));
    %pr=0
end

 if ((al>=al2)&(al<=al3))|((al>=-pi-al3)&(al<=-pi-al2))
    rr=a*sqrt(1+tan(al).*tan(al));
end

 if ((al>=al3)&(al<=al4))|((al>=-pi-al4)&(al<=-pi-al3))
    rr=a*sqrt(1+1./(tan(al).*tan(al)));
end

 if (((al>=al4)&(al<=al5)))...
        |(((al>=pi-al5)&(al<=pi))|((al>=-pi)&(al<=-pi-al4)))
        rr=b*sqrt(1+tan(al).*tan(al));%3*
end

 if ((al>=al5)&(al<=al6))|((al>=pi-al6)&(al<=pi-al5))
    rr=a*sqrt(1+1./(tan(al).*tan(al)));
end

 if ((al>=al6)&(al<=al7))|((al>=pi-al7)&(al<=pi-al6))
    rr=a*sqrt(1+tan(al).*tan(al));
end
 if (al>=al7)&(al<=pi-al7)
    rr=b*sqrt(1+1./(tan(al).*tan(al)));%3*
end
```

4.6. u0lapt1.m

```
function r1=u0lapt1(x,y);
%r1=1;
r1=0.25*(x^2+y^2); % Solución analítica
```

4.7. pdecirculopoisson549.m

```
function pdemodel tic [pde_fig,ax]=pdeinit; pdetool('appl_cb',1);
set(ax,'DataAspectRatio',[1 1 1]);
set(ax,'PlotBoxAspectRatio',[1.5 1 1]); set(ax,'XLim',[-1.5 1.5]);
set(ax,'YLim',[-1 1]); set(ax,'XTickMode','auto');
set(ax,'YTickMode','auto');
% Geometry description:
pdeellip(0,0,1,1,... 0,'Poissonunitario');
set(findobj(get(pde_fig,'Children'),'Tag','PDEEval'),'String','Poissonunitario')
% Boundary conditions:
pdetool('changemode',0) pdesetbd(4,... 'dir',... 1,... '1',...
'(x.*x+y.*y)*(0.25)') pdesetbd(3,... 'dir',... 1,... '1',...
'(x.*x+y.*y)*(0.25)') pdesetbd(2,... 'dir',... 1,... '1',...
'(x.*x+y.*y)*(0.25)') pdesetbd(1,... 'dir',... 1,... '1',...
'(x.*x+y.*y)*(0.25)')
% Mesh generation:
setappdata(pde_fig,'Hgrad',1.3);
setappdata(pde_fig,'refinemethod','regular'); pdetool('initmesh')
pdetool('refine')
% PDE coefficients:
pdeseteq(1,... '1.0',... '0.0',... '-1',... '1.0',... '0:10',...
'0.0',... '0.0',... '[0 100]') setappdata(pde_fig,'currparam',...
['1.0';... '0.0';... '-1 ';... '1.0'])
% Solve parameters:
setappdata(pde_fig,'solveparam',...
str2mat('0','1548','10','pdeadworst',...
'0.5','longest','0','1E-4','','fixed','Inf'))
% Plotflags and user data strings:
setappdata(pde_fig,'plotflags',[1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0 0
1]); setappdata(pde_fig,'colstring','');
setappdata(pde_fig,'arrowstring','');
setappdata(pde_fig,'deformstring','');
setappdata(pde_fig,'heightstring','');
% Solve PDE:
pdetool('solve') tiempoPDETOOL=toc
```

4.8. pdecruzpoisson457.m

```
mfunction pdemodel tic [pde_fig,ax]=pdeinit; pdetool('appl_cb',1);
set(ax,'DataAspectRatio',[1 1.5 1]);
set(ax,'PlotBoxAspectRatio',[1 0.66666666666666663 2]);
set(ax,'XLim',[-0.5 0.5]); set(ax,'YLim',[-0.5 0.5]);
set(ax,'XTickMode','auto'); set(ax,'YTickMode','auto');

% Geometry description:
pderect([-0.45000000000000001 0.45000000000000001
0.14999999999999999 -0.14999999999999999],'R1');
pderect([-0.14999999999999999 0.14999999999999999
-0.45000000000000001 0.45000000000000001],'R2');
set(findobj(get(pde_fig,'Children'),'Tag','PDEEval'),'String','R1+R2')

% Boundary conditions:
pdetool('changemode',0) pdetool('removeb',[6 9 12 15 ]);
pdesetbd(12,... 'dir',... 1,... '1',... '0.25*(x.*x+y.*y)')
pdesetbd(11,... 'dir',... 1,... '1',... '0.25*(x.*x+y.*y)')
pdesetbd(10,... 'dir',... 1,... '1',... '0.25*(x.*x+y.*y)')
pdesetbd(9,... 'dir',... 1,... '1',... '0.25*(x.*x+y.*y)')
pdesetbd(8,... 'dir',... 1,... '1',... '0.25*(x.*x+y.*y)')
pdesetbd(7,... 'dir',... 1,... '1',... '0.25*(x.*x+y.*y)')
pdesetbd(6,... 'dir',... 1,... '1',... '0.25*(x.*x+y.*y)')
pdesetbd(5,... 'dir',... 1,... '1',... '0.25*(x.*x+y.*y)')
pdesetbd(4,... 'dir',... 1,... '1',... '0.25*(x.*x+y.*y)')
pdesetbd(3,... 'dir',... 1,... '1',... '0.25*(x.*x+y.*y)')
pdesetbd(2,... 'dir',... 1,... '1',... '0.25*(x.*x+y.*y)')
pdesetbd(1,... 'dir',... 1,... '1',... '0.25*(x.*x+y.*y)')

% Mesh generation:
setappdata(pde_fig,'Hgrad',1.3);
setappdata(pde_fig,'refinemethod','regular'); pdetool('initmesh')
pdetool('refine')

% PDE coefficients:
pdeseteq(1,... '1.0',... '0.0',... '-1',... '1.0',... '0:10',...
'0.0',... '0.0',... '[0 100]') setappdata(pde_fig,'currparam',...
['1.0';... '0.0';... '-1 ';... '1.0'])

% Solve parameters:
setappdata(pde_fig,'solveparam',...
str2mat('0','1224','10','pdeadworst',...
```

```
'0.5','longest','0','1E-4','','fixed','Inf'))

% Plotflags and user data strings:
setappdata(pde_fig,'plotflags',[1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0 0
1]); setappdata(pde_fig,'colstring','');
setappdata(pde_fig,'arrowstring','');
setappdata(pde_fig,'deformstring','');
setappdata(pde_fig,'heightstring','');

% Solve PDE:
pdetool('solve') tiempoPDETOOLBOX=toc
```

Bibliografía

[1] A. Grebennikov, A. Cortés, A. Espíndola, *Transformada de la derivada parcial de una función en un dominio acotado en el plano*, Revista Iberoamericana de Ciencias, Vol. 1, No. 4. Sept 2014, pp 169-176. ISSN: 2334-2501.

[2] Alexandre Grebennikov, On Theorical Foundation of p-version of General Ray Method for solution of the Dirichlet Boundary Value Problems for Poisson Equation in Plane Domanins. International Journal of Scientific and Innovative Mathematical Research, Volume 2, Issue 12, Decembre 2014, pp 960-965. ISSN 2347-307X (Print) & ISSN 2347-3142 (Online)

[3] Andrei D. Polyanin *Handbook of Linear Partial Differential Equations for Engineers and Scientists* Chapman Hall CRC, Boca Raton-Florida 2001.

[4] Gordon D. Smith *Numerical Solution of Partial Differential Equations*, Clarendon Press, Oxford, Second edition, 1978

[5] Peter Kattan, Matlab Guide to Finite Elements: an interactive approach. Springer-Verlag, Berlin, 2008

[6] Gennadiy Nikishkov, *Programming Finite Elements in Java* Springer-Verlag, London, 2010.

[7] Dean G. Duffy, *Transform Methods for Solving Partial Differential Equations* Chapman Hall-CRC, Boca Raton, 2004

[8] The MathWorks *Partial Differential Equation: User´s guide* Computer Solutions Europe AB 2015

[9] M. Necati Ozisik, *Heat conduction.* John Wiley & Sons, 1980

[10] Alexander D. Poularikas, *The Transforms and Applications Handbook* Second Edition, CRC, Boca Raton 2000

[11] Kak, A. C. and M. Slaney, *Principles of Computerized Tomographic Imaging*, New York, NY, IEEE Press, 1988

[12] Helgason Sigurdur. *The Radon Transform*, Birkhauser, Boston-Basel-Berlin, 1999

[13] Oluleke Oluwole, *Finite Element Modeling for Materials Engineers Using MATLAB*, Springer-Verlag, London 2011

[14] Garrett Birkhoff *The Numerical Solution of Elliptic Equations* SIAM, Philadelphia, 1972

[15] W. Hackbusch *Elliptic Differential Equations Theory and Numerical Treatment* Springer-Verlag, Berlin 2003

[16] A. Grebennikov , E.A. Romano Castillo , S. Reyes Mora, R. Paredes Jaramillo. *Application of Radon Transform for Explicit Solution of Boundary Value Problems for Elliptic PDE.* Applied Math IV, Cuarto Congreso Internacional en Matemáticas Aplicadas. Memorias en extenso, IPN, México, 2008, CDROM, pp. 1-8.

[17] A. Grebennikov, R. Paredes Jaramillo. *Program Realization of Gr-Algorithm and It's Numerical Experimental Comparison on Rapidity with PDEmodel Program Of Matlab System.* Computer Modeling 2008. Proceedings of the International Scientific-Technical Conference. Saint -Petersburg Technical University Press, Russia, 2008, pp. 64 - 67. ISBN: 978-5-7422-1874-6.