UNIVERSIDAD SAN FRANCISCO DE QUITO

Desarrollo de Software para análisis bidimensional de Transferencia de Calor en estado estable mediante el Método de los Elementos Finitos

David Roberto Escudero Guevara

Tesis de Grado presentada como requisito para la obtención del título de Ingeniero Mecánico

Quito, Enero 2008

DEDICATORIA

A mi familia por todo el amor y el cariño. A mis padres, por los años de esfuerzo y sacrificio para darme lo mejor, por apoyarme en todas mis decisiones, hayan sido estas buenas o malas, por enseñarme que cuando uno se propone algo en la vida lo consigue, por hacerme ver que soy capaz de hacer cualquier cosa en este mundo, si lo hago con mucho esfuerzo y dedicación.

AGRADECIMIENTOS

Quisiera agradecer a Víctor Guerrero por su gran ayuda y apoyo en la realización de este trabajo. A Valeria por ser un apoyo incondicional y por siempre estar ahí, en los buenos y malos momentos. A Eduardo y a Cristina por brindarme una ayuda desinteresada y facilitar la terminación de este proyecto.

RESUMEN

En este trabajo se ha desarrollado, con fines didácticos y de investigación, un programa computacional para analizar problemas bidimensionales de transferencia de calor en estado estable mediante el método de los elementos finitos. La codificación del programa se realizó utilizando el lenguaje Java y las herramientas gratuitas de desarrollo NetBeans 5.0 y Java 3D. Con estas herramientas se construyó una interfaz gráfica de usuario amigable y de fácil manejo. Esta interfaz permite dibujar la geometría de los sistemas a estudiar, discretizarlos, imponer las condiciones de frontera, resolver los sistemas de ecuaciones y desplegar los resultados gráfica y numéricamente. Para discretizar los dominios a analizar, el programa permite utilizar mallados estructurados y no estructurados con elementos triangulares o rectangulares. El mallado no estructurado se realiza mediante el método de avance frontal. La solución de los sistemas de ecuaciones resultantes se realiza empleando las librerías gratuitas de cálculo numérico colt y mtj. Además, el programa permite que en un futuro se puedan incorporar nuevos tipos de análisis, opciones de cálculo, tipos de mallado, etc. Finalmente, para validar los resultados se analizó un conjunto de problemas utilizando Algor y Cosmos Works, así como también se comparó los resultados con la solución analítica. La comparación de resultados obtenidos permite afirmar que con el programa desarrollado se puede obtener resultados similares a los de los programas comerciales considerados.

ABSTRACT

In this work, a finite element software for analysis of two–dimensional heat transfer steady state problems has been developed, for didactic and research purposes. The program was coded using Java and the free development toolkits NetBeans 5.0 and Java 3D. These tools were used to build a friendly and easy to handle graphic user's interface. This interface is used for drawing and meshing the geometry of the systems to be studied, imposing the boundary conditions, solving the resulting systems of equations, and presenting the results graphically and numerically. For meshing the problems to be analyzed, the program offers the option of using structured and unstructured meshes, using triangular or rectangular elements. The unstructured meshing is carried out by the frontal advance method. The solution of the resulting algebraic equations is found using the free distribution toolboxes for numerical calculation colt and mtj. The program has been also coded so that new types of analyses, and calculation and meshing options, can be implemented in the future. Finally, to validate the results, a selected set of problems was analytically solved and analyzed using Algor and Cosmos Works. The comparison confirms that the results obtained with the program developed are similar to those obtained with the commercial software.

CONTENIDO

CONTENIDO DE FIGURAS

CONTENIDO DE TABLAS

INTRODUCCIÓN

Hoy en día existen en el mercado un sinnúmero de paquetes computacionales que permiten resolver una gran variedad de problemas de ingeniería con el método de los elementos finitos. Para citar algunos ejemplos, se tiene Nastran, Cosmos Works, Abaqus, Algor, Ansys, etc. Estos programas tienen la característica de ser diseñados por grandes compañías, que invierten mucho dinero en investigación y desarrollo, por lo que el costo para adquirir estos paquetes y sus respectivas licencias es muy alto como para que un estudiante los pueda comprar.

Por esta razón, se planteó la idea de crear un software de elementos finitos que resuelva problemas de ingeniería tales como los que involucran esfuerzos estáticos

lineales y transferencia de calor en estado estable. Uno de los aspectos importantes por los cuales se hizo este planteamiento es que este software, a diferencia de los grandes paquetes computacionales comerciales, se lo puede desarrollar sin una gran inversión de dinero, y no se requiere invertir por la compra de una licencia para su uso. En segundo lugar, partiendo de este programa inicial y dada la arquitectura del mismo se puede aumentar cada vez más opciones de cálculo, tipos de análisis, mejoras en el tipo de mallado, mejor presentación estética, etc. Es decir, no es un programa estático sino más bien un programa dinámico que puede ser mejorado y que a la larga puede ser una herramienta útil y fácil de usar al momento de requerir de análisis de relativa complejidad y que el programa pueda resolver.

Este programa es de mucha utilidad en el área didáctica, pues es un programa no muy complejo en su estructura, es de fácil manejo y de fácil enseñanza, lo que permitiría a profesores y alumnos tener una herramienta básica para la resolución de problemas en un principio estático lineales y de transferencia de calor para elementos tipo placa y elementos 2D, pero con la posibilidad de que un futuro se puedan resolver otro tipo de problemas más complejos y con otro tipo de elementos.

Finalmente, los objetivos planteados para el presente trabajo están muy relacionados con las razones explicadas en los párrafos anteriores, ya que estas resumen la importancia del proyecto y de este trabajo. Dichos objetivos son:

- Conocer, aprender y entender el funcionamiento y programación de la versión 1.0 del programa Placa.

- Desarrollar nuevas subrutinas de programación que permitan resolver problemas de transferencia de calor usando el método de los elementos finitos y acoplarlo a la primera versión de Placa.

- Desarrollar una nueva interfaz gráfica de usuario que contenga los nuevos requerimientos para el análisis planteado, y relacionar esta nueva interfaz con la de la primera versión de Placa.

CAPÍTULO 1

INTRODUCCIÓN AL MÉTODO DE LOS ELEMENTOS FINITOS

En este capítulo se tratan temas generales relacionados con el método de los elementos finitos, su definición, como funciona el análisis mediante este método y cuáles son los pasos a seguir para realizar un óptimo análisis. También se presenta una breve comparación con otros métodos matemáticos usados para la resolución de problemas de ingeniería y se muestra como se aplica este método a la resolución de problemas de transferencia de calor.

1.1. Generalidades del método de los elementos finitos

El método de los elementos finitos (MEF) es una técnica de análisis numérico que permite encontrar soluciones aproximadas a problemas de ingeniería formulados como problemas de valor en la frontera. Estos son problemas dados en términos de un determinado número de variables dependientes que se relacionan por una o varias ecuaciones diferenciales las cuales se deben satisfacer siempre, y dentro de un dominio específico de variables independientes. Estas ecuaciones cumplen además ciertas condiciones en la frontera del dominio.

Para ilustrar la forma en que opera el MEF se considerará la Figura 1.1, tomada del libro *Fundamentals of Finite Element Analysis* de David Hutton [3]. En esta figura, el volumen exterior representa el dominio del problema a resolver. La variable $\Phi(x, y)$ es la variable dependiente, conocida también como *variable de campo,* cuyos valores se desea encontrar en cada punto $P(x, y)$ del dominio, en el cual se satisface la ecuación diferencial que rige el problema. Las variables independientes en este caso son las coordenadas espaciales x e y.

El MEF divide al dominio de variables del problema en subdominios a través de elementos de geometría conocida. A este proceso de división del dominio se le denomina discretización o mallado. En cada uno de los elementos resultantes del mallado, la ecuación diferencial que rige el problema es modelada a través de un conjunto de ecuaciones algebraicas que usualmente se presentan en formato matricial de la siguiente forma: [k]{u}={f}, donde [k] es la matriz de rigidez del elemento, {u} es el denominado vector de desplazamientos nodales, mismo que engloba los valores de la variable de campo que se desea calcular, y {f} es el vector de cargas nodales.

Los elementos en que se ha dividido al dominio tienen nodos asociados, tales como los especificados con los números 1, 2, 3 en la Figura 1.1. Estos nodos pueden ser interiores o exteriores. Los interiores se encuentran dentro del elemento mientras que los exteriores se encuentran en su frontera. Los nodos representan los puntos en donde se desea calcular explícitamente el valor de las variables de campo.

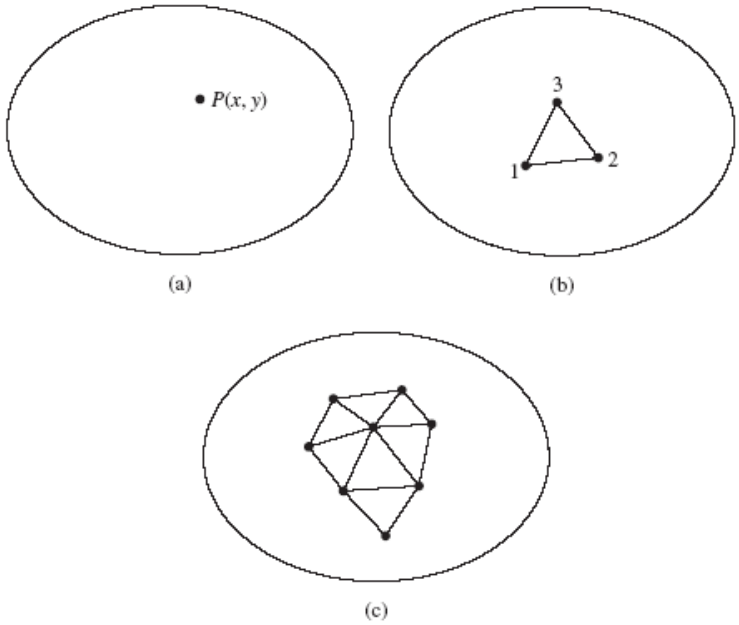


Figura 1.1. (a) Dominio bidimensional con la variable de campo Φ(*x*, *y*). (b) Un elemento finito de tres nodos definido en el dominio. (c) Elementos extra mostrando una malla parcial de elementos finitos en el dominio. [3]

Para calcular los valores de las variables de campo en los nodos del problema es necesario resolver el denominado sistema global de ecuaciones algebraicas: [K]{U}={F}, donde [K] es la matriz de rigidez global del sistema analizado, {U} es el vector de desplazamientos globales y {F} es el vector de cargas globales. Para obtener este sistema se debe ensamblar de manera apropiada los sistemas de ecuaciones asociados con los elementos en que se discretizó el dominio del problema. Además, antes de resolverlo, se debe imponer las condiciones de frontera. Finalmente, vale la pena indicar que para el tipo de problemas con el que se va a tratar en este trabajo, la [K] es una matriz cuadrada, simétrica, singular, dispersa y bandada.

Una vez obtenidos los valores de las variables de campo en los nodos asociados con los elementos finitos, se requiere calcular también los valores de las variables de campo pero en otros puntos del elemento. Para esto se lleva adelante un proceso de interpolación, mismo que se efectúa con la ayuda de una serie de funciones conocidas

como *funciones de interpolación* o *funciones de forma*. Con estas funciones se puede obtener una aproximación de los valores de las variables de campo en todos los puntos del dominio.

La ecuación 1.1 representa la relación entre las funciones de interpolación (N_1, N_2, N_3) y los valores de la variable de campo (Φ_1, Φ_2, Φ_3) para un elemento finito de tres nodos. Generalmente estas funciones de interpolación están dadas en forma de polinomios y son derivadas considerando que las variables dependientes asociadas con el problema que se desea resolver deben satisfacer ciertas condiciones en los nodos.

$$\Phi(x, y) = N_1(x, y)\Phi_1 + N_2(x, y)\Phi_2 + N_3(x, y)\Phi_3 \qquad (1.1)$$

Como se observa en la Figura 1.2, no toda el área de un dominio bidimensional se puede cubrir con los elementos finitos en que se discretiza. Esto implica que si a un problema de ingeniería se le aplica el MEF, la respuesta que se obtenga va a ser necesariamente una aproximación de la solución exacta. Dicha aproximación puede ser mejorada hasta el punto en donde las soluciones obtenidas mediante el MEF converjan hacia la solución exacta, pudiéndose incluso alcanzar esta solución, al menos en ciertos puntos del dominio.

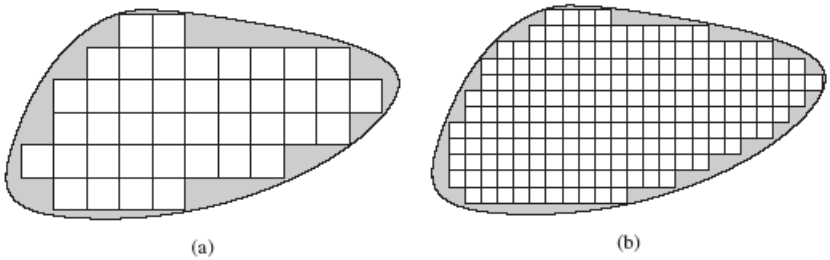


Figura 1.2. (a) Área discretizada parcialmente con elementos tipo cuadrado. (b) La misma área pero discretizada con elementos más pequeños de la misma geometría. [3]

Para ilustrar lo anteriormente dicho se propone un ejemplo tomado del libro de David Hutton *Fundamentals of Finite Element Analysis* [3]. En este ejemplo se quiere encontrar el desplazamiento del extremo libre de una barra de sección transversal variable sometida a una fuerza F.

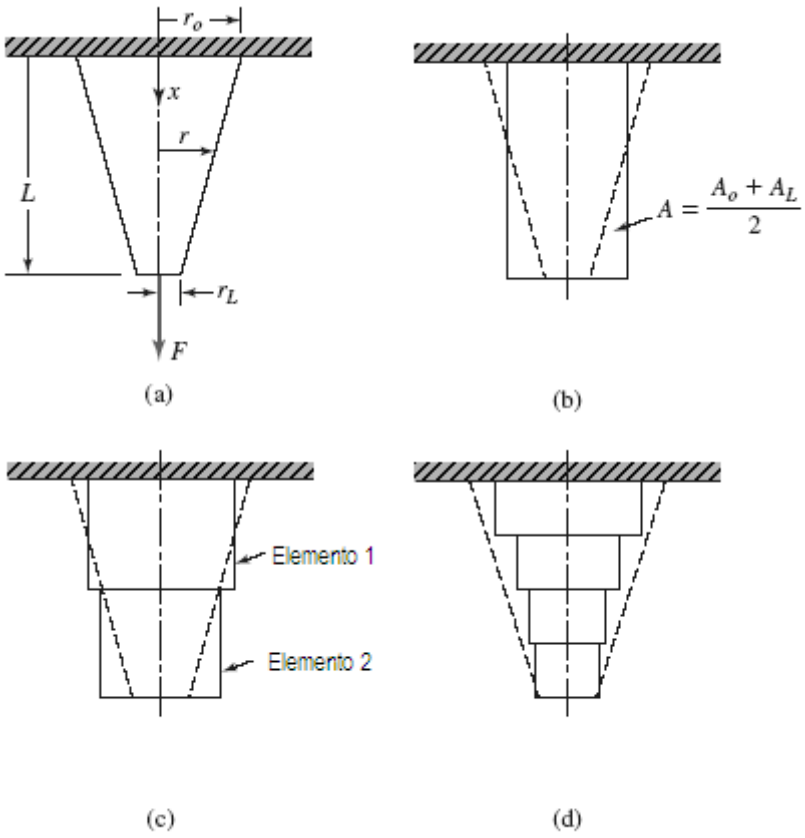


Figura 1.3. (a) Barra sujeta a una carga de tracción F. (b) Barra como un elemento de área uniforme. (c) Barra modelada como dos elementos finitos del mismo largo. (d) Barra modelada con cuatro elementos finitos del mismo largo. En (b) y (c) el área de cada elemento es el promedio del área de la barra a lo largo de la longitud del elemento. [3]

Utilizando el MEF y conociendo la solución exacta se tienen los resultados que se muestran en la Figura 1.4. En esta figura se puede apreciar como la solución obtenida por el MEF se aproxima a la solución exacta a medida que el número de elementos del mallado aumenta. En un problema de ingeniería típico, el número de elementos finitos que se necesita para encontrar una solución suficientemente aproximada a la solución exacta es relativamente elevado y se necesita resolver grandes sistemas de ecuaciones. De allí que el empleo del MEF demande su implementación computacional. Adicionalmente, esta implementación computacional y las amplias prestaciones del software moderno facilitan el ingreso de los datos del problema y la visualización de los resultados.

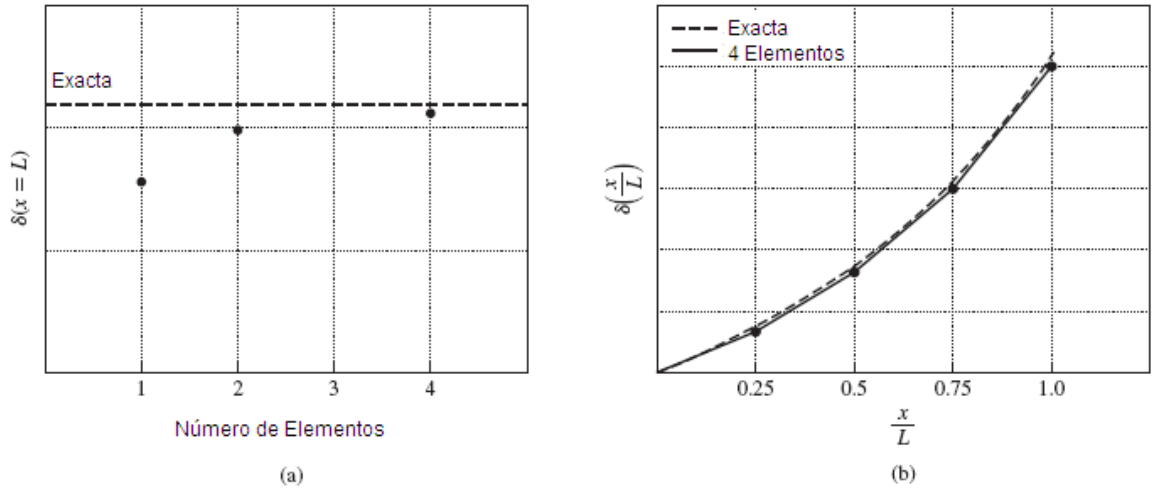


Figura 1.4. (a) Desplazamiento a la distancia x = L de la barra sujeta a tracción. (b) Comparación de la solución exacta con la solución obtenida mediante el MEF para una discretización con 4 elementos. [3]

1.2. Análisis por el método de los elementos finitos

Los pasos a seguir para realizar un análisis mediante el MEF son:

Pre-procesamiento: en esta etapa se debe realizar lo siguiente:

- Crear y discretizar el dominio del problema en elementos finitos; esto es subdividir en nodos y elementos.
- Asumir una función de forma que represente el comportamiento físico de un elemento; esto es, se asume que una función continua representa la solución de un elemento.
- Desarrollar las ecuaciones para cada elemento.
- Ensamblar las ecuaciones y encontrar la matriz global del sistema.
- Aplicar las condiciones de borde, las condiciones iniciales y las cargas.[8]

Solución: en esta etapa se debe:

- Resolver el conjunto de ecuaciones algebraicas lineales o no lineales simultáneamente para encontrar los valores nodales de las variables de campo, sean estas desplazamientos para el caso de mecánica de sólidos o temperaturas para el caso de transferencia de calor.[8]

Post-procesamiento:

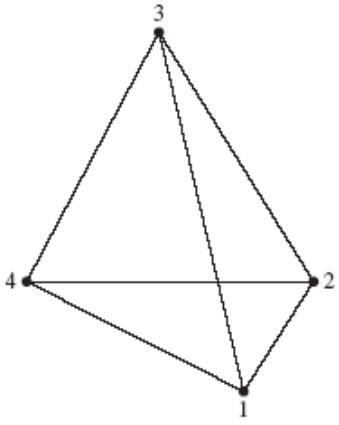
- Se obtiene información adicional importante, como puede ser esfuerzos y flujos de calor.[8]

1.2.1. Tipos de elementos finitos

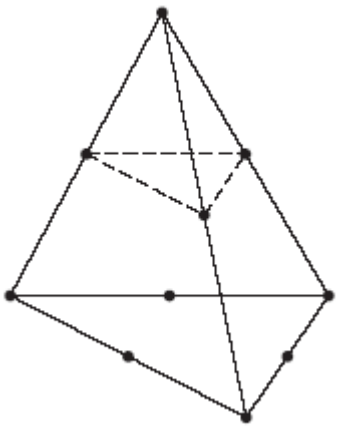
Como se dijo anteriormente, el mallado se lo puede hacer con una serie de elementos de geometría conocida, por lo que existen algunos tipos de elementos que se pueden usar. En la Figura 1.5 se muestran elementos triangulares y rectangulares, que se pueden utilizar para discretizar dominios bidimensionales. También se muestran elementos tetraédricos y tipo ladrillo, que se pueden emplear para mallar dominios tridimensionales. Cada elemento tiene un cierto número de nodos relacionados, que pueden estar dispuestos tanto en la frontera del elemento como en su interior. Así, en la Figura 1.5 se muestran elementos triangulares con 3, 6 y 10 nodos. El número de nodos que tiene un elemento en particular afecta la forma de las funciones de interpolación que se emplean y cambia el número de ecuaciones algebraicas que se asocian con ese elemento y por ende el número de operaciones a realizar y el tiempo que toma resolver un problema determinado.

- Elementos triangulares

- Elementos tetraédricos



- Elementos rectangulares



- Elementos tipo ladrillo

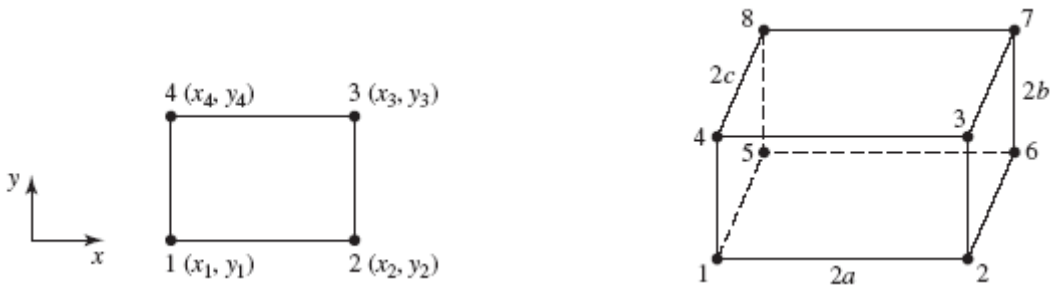


Figura 1.5. Distintos tipos de elementos usados para el análisis por elementos finitos. [3]

1.2.2. Resolución de problemas por el método de los elementos finitos

El método de los elementos finitos permite resolver una serie de complejos problemas de ingeniería de una forma relativamente simple. Un paso de vital importancia en el desarrollo del análisis por el MEF es encontrar la matriz de rigidez de cada elemento finito que permite modelar la ecuación diferencial que rige el problema mediante una serie de ecuaciones algebraicas. Esta matriz de rigidez contiene la información de la geometría y del comportamiento del material e indica la resistencia del elemento cuando es sometido a cargas mecánicas, térmicas, etc.

Para derivar la matriz de rigidez se utilizan varios métodos, mismos que dependen de la condición del problema. Algunos de estos métodos son:

- El teorema de Castigliano
- El método de equilibrio estático
- El principio de la energía potencial mínima

Para sistemas de más de un elemento se debe llevar a cabo un proceso de ensamble de las matrices de rigidez de los distintos elementos. Los siguientes pasos son los recomendados en el proceso de resolver un problema por el MEF:

1. *Discretizar el dominio del problema:* Consiste en dividir la región o dominio del problema en elementos finitos, mismos que pueden ser de las formas explicadas anteriormente.

2. *Seleccionar las funciones de interpolación:* El segundo paso consiste en numerar los nodos para cada elemento de la región. De esta forma se selecciona las funciones de interpolación y se puede representar las variaciones de la variable de campo dentro del elemento finito.

3. *Hallar las propiedades del elemento:* Consiste en establecer una serie de ecuaciones matriciales para cada elemento, involucrando sus propiedades. Existen tres tipos de enfoques para realizar este proceso: el enfoque directo, el enfoque variacional, o el enfoque de residuos ponderados.

4. *Ensamblar las ecuaciones matriciales de cada elemento para obtener el sistema de ecuaciones:* En esta etapa del proceso se ensamblan cada una de las matrices de los elementos en una matriz global que contiene todas las propiedades del sistema. La base para el procedimiento de ensamblaje parte de que en un nodo en donde los elementos se interconectan, el valor de la variable de campo es el mismo para cada elemento que comparte ese nodo.

5. *Imponer las condiciones de frontera:* Antes de que el sistema de ecuaciones esté listo para ser resuelto, se lo debe modificar para tomar en cuenta las condiciones de frontera del problema. En esta etapa se imponen los valores nodales conocidos de las variables dependientes.

6. *Resolver el sistema de ecuaciones***:** El proceso de ensamblado resulta en un sistema de ecuaciones simultáneas que se resuelve para obtener los valores nodales desconocidos del problema. Si el problema describe un estado estable o en equilibrio, entonces se debe resolver un sistema de ecuaciones algebraicas lineales o no lineales. Si el problema es inestable, las incógnitas nodales son una función del tiempo, y se debe resolver un sistema de ecuaciones diferenciales ordinarias lineales o no lineales.

7. *Calcular variables adicionales:* Muchas veces se usa la solución del sistema de ecuaciones para calcular otros parámetros importantes. Por ejemplo, en un problema estructural las incógnitas nodales son las componentes del desplazamiento. A partir de estos desplazamientos se calculan los esfuerzos y deformaciones de los elementos.

1.3. Comparación del MEF con el método de las diferencias finitas

El método de las diferencias finitas es otra técnica de análisis numérico empleada para la resolución de problemas de ingeniería que están regidos por una o varias ecuaciones diferenciales. En el método de las diferencias finitas las derivadas parciales de la ecuación diferencial que rige el problema son reemplazadas por expresiones algebraicas, por lo que su resolución se reduce a encontrar una serie de ecuaciones algebraicas y su respectiva solución. Para poder lograr esta aproximación de las derivadas parciales, se debe primero discretizar el dominio del problema mediante una malla, es decir, establecer puntos (nodos) en los cuales se desea obtener el valor de la variable desconocida. Posteriormente, con la formulación de las series de Taylor se obtiene expresiones para cada una de las derivadas, con lo cual se puede obtener un sistema de ecuaciones lineales que resuelven el problema y calculan la variable deseada.

La diferencia entre este método y el método de los elementos finitos radica en que en el método de los elementos finitos la variación de las variables de campo en el dominio es una parte integral del proceso. Con el uso de funciones de interpolación la variación de las variables de campo entre elementos permite calcular los valores de dichas variables en todo el dominio del problema. Por otro lado, en el método de las diferencias finitas no se aplica este criterio puesto que la variable de campo no se calcula para todo el dominio sino solo para puntos específicos.

Otra diferencia importante entre estos dos métodos de análisis numérico radica en el hecho que el método de las diferencias finitas modela las ecuaciones diferenciales que rigen el problema y utiliza métodos de integración numérica para encontrar la solución en puntos específicos. Por su parte, el método de los elementos finitos modela todo el dominio del problema y utiliza principios de la física para desarrollar una serie de ecuaciones algebraicas para encontrar soluciones aproximadas del problema.

Pero así como existen diferencias, también existen ciertas similitudes entre estas dos técnicas o métodos. La primera similitud existente es que en el método de las diferencias finitas los puntos en donde se realizan las integraciones son análogos a los nodos del MEF, pues es en estos puntos en donde se calcula los valores de las variables de campo. La segunda similitud radica en la convergencia de la solución con la solución exacta. En el método de las diferencias finitas la solución empieza a converger con la solución exacta a medida que el paso de integración se reduce, esto es, cuando la distancia que separa los puntos es menor y la malla tiene más nodos. En el MEF esto sucede cuando la malla de elementos es mejorada o refinada; es decir, cuando el número de elementos de la malla se incrementa

1.4. Aplicaciones del método de los elementos finitos

El MEF es un método muy útil para la resolución de varios problemas de ingeniería. Algunas de sus aplicaciones incluyen:

- Problemas de transferencia de calor
- Mecánica de sólidos
- Análisis de vibraciones
- Electromagnetismo
- Mecánica de fluidos

A continuación se muestra una forma de aplicación del método de los elementos finitos para el análisis de problemas de transferencia de calor utilizando elementos triangulares y elementos rectangulares implementados en programas computacionales. También se describe el desarrollo de un programa propio para el análisis de este tipo de problemas.

1.5. El MEF y los problemas de transferencia de calor

Como se dijo anteriormente, el método de los elementos finitos permite resolver una serie de problemas de ingeniería en diversos campos. Uno de estos tipos de problemas es el de transferencia de calor.

La transferencia de calor es el intercambio de energía térmica ocurrido por una diferencia de temperaturas ya sea dentro de un medio o entre medios. Siempre que exista una diferencia de temperatura va a existir una transferencia de calor. Existen varios mecanismos de transferencia de calor:

- Conducción: Es el término utilizado para la transferencia de calor que se ocasiona cuando los medios involucrados son sólidos o fluidos. A la conducción también se la relaciona con transferencia de energía a nivel atómico, es decir la conducción es una transferencia de energía desde partículas con un nivel mayor de energía hacia partículas de menor valor energético debido a las interacciones existentes entre las partículas de una sustancia. La ecuación que determina la conducción es la siguiente:

$$q'' = -k\frac{dT}{dx} \tag{1.2}$$

Donde q es el flujo de calor (W/m^2), k es la conductividad térmica (W/mK), asociada a cada material y que depende de la temperatura a la que se encuentre el material y dT/dx es el gradiente de temperatura (K/m).

- **Convección:** Es la transferencia de calor que se produce entre una superficie y un fluido que se encuentran a distintas temperaturas. Dentro de este mecanismo se distinguen dos tipos de convección. La *convección forzada*, la cual se da cuando el flujo es producido por un componente externo, como un ventilador. La *convección natural*, la cual se da cuando el fluido se mueve por causas naturales, como el efecto de flotación, el cual se manifiesta con la subida del fluido caliente y el descenso del fluido frío.

La ecuación de convección es la siguiente:

$$q'' = h\left(T_s - T_\infty\right) \tag{1.3}$$

donde q'' es el flujo de calor convectivo (W/m^2), h es el coeficiente de transferencia de calor (W/m^2K), y depende de la geometría de la superficie y del tipo de fluido en movimiento. T_s es la temperatura de la superficie y T_∞ es la temperatura del fluido que interactúa en el proceso.

- Radiación: Es la transferencia de calor en forma de ondas electromagnéticas entre dos superficies que se encuentran a distintas temperaturas. Esta transferencia de calor se puede dar sin la presencia de un medio material.

En el proceso de radiación existen dos superficies involucradas: la una, que es conocida como el objeto radiador, es la superficie que irradia la energía térmica, y la otra superficie es la que absorbe esta energía térmica. Por esta razón, cada elemento tiene tasas relacionadas con la cantidad de energía liberada y la cantidad de energía absorbida. La ecuación de radiación es la siguiente:

$$q'' = \varepsilon\sigma\left(T_s^{\,4} - T_{sur}^{\,4}\right) \tag{1.4}$$

donde ε es la emisividad de la superficie, σ es la constante de Stefan-Boltzmann ($5.67x10^{-8}$ W/m^2K^4), T_s es la superficie la superficie y T_{sur} es la temperatura del entorno que rodea a la superficie.

En este trabajo se desarrollará en Java un programa que permita resolver problemas bidimensionales de transferencia de calor en estado estable. Para esto se tomará en cuenta la presencia del fenómeno de conducción y del fenómeno de convección, y el fenómeno de radiación no se lo considera.

Los problemas que involucran transferencia de calor se los cataloga dentro de los problemas con valor en la frontera los cuales están regidos por una ecuación diferencial, la cual depende y está conformada por los diferentes fenómenos explicados anteriormente de transferencia de calor.

Para comenzar el análisis por el método de los elementos finitos se debe definir la naturaleza del problema. Es decir, se debe definir si se tratará con conducción unidimensional, conducción bidimensional, conducción y convección, conducción, convección y generación interna de calor. Una vez definida la naturaleza del problema se expresa la ecuación diferencial que lo rige. La derivación de la ecuación diferencial que expresa la presencia de los fenómenos a tratar en este trabajo se encuentra en el libro *Fundamentals of Finite Element Analysis* de David Hutton [3]. A continuación se presenta la ecuación diferencial ya derivada:

$$k_x t \frac{\partial^2 T}{\partial x^2} + k_y t \frac{\partial^2 T}{\partial y^2} + Qt = 2h\left(T - T_a\right) \tag{1.5}$$

Los primeros dos términos representan la conducción en la dirección *x* y en la dirección *y*, el tercer término representa la generación interna de calor y el último término representa el fenómeno de convección. Cabe recordar que los problemas que van a ser analizados por el programa a codificar son problemas en estado estable, es decir, problemas que no son variables con el tiempo. La variable t representa el espesor de la geometría a analizar, la variable Q es el valor en W/m^3, cuando la geometría analizada presenta generación interna de calor y finalmente la variable T_a es la temperatura del fluido que produce la convección.

Una vez definida la ecuación diferencial que rige el sistema, el siguiente paso es definir una geometría, la cual va a ser el dominio del problema a resolver. Se establecen en dicha geometría las condiciones de borde para la resolución del problema; para los problemas de transferencia de calor normalmente las condiciones de borde tienen que ver con la designación de temperaturas iniciales, ya que la variable de campo en este tipo de problemas es precisamente la temperatura.

Cuando se ha definido la geometría y las condiciones de borde, el siguiente paso es decidir que tipo de elemento se va a utilizar para la resolución del problema y encontrar las matrices de rigidez para cada uno de los elementos escogidos y los términos de fuerzas. De esta manera se encuentran todos los términos que conforman la ecuación [K] {T}= {F}. Para este trabajo se ha elegido dos tipos de elementos: el elemento triangular de tres nodos y el elemento rectangular de 4 nodos.

1.5.1. Elementos triangulares

Los elementos triangulares se dividen de acuerdo a las funciones de interpolación inherentes a cada tipo de elemento, por esta razón existen tres tipos usuales de elementos triangulares, que están representados en la Figura 1.6. Estos elementos tienen 3, 6 y 10 nodos, lo cual resulta en funciones de interpolación lineales, cuadráticas y cúbicas respectivamente.

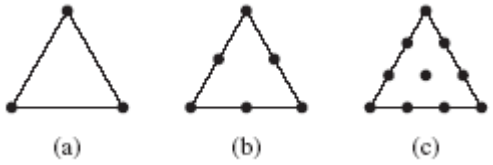


Figura 1.6. (a) Elemento triangular de 3 nodos. (b) Elemento triangular de 6 nodos. (c) Elemento triangular de 10 nodos. [3]

Para este trabajo se ha escogido utilizar el elemento triangular de tres nodos, para el cual se van a deducir las funciones de interpolación y la matriz de rigidez que permitan resolver problemas bidimensionales de conducción de calor en estado estable. Primero se determinan las funciones de interpolación para el elemento triangular de tres nodos (Figura 1.7), para la cual se considera que cada nodo tiene un solo grado de libertad dado por la temperatura del cuerpo analizado en ese punto.

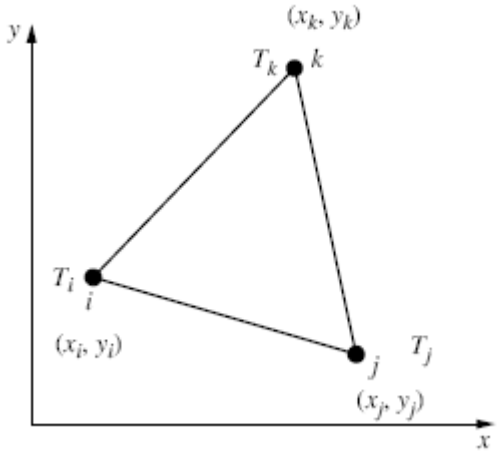


Figura 1.7. Elemento triangular. [5]

La función polinomial de la temperatura para el elemento triangular de 3 nodos es:

$$T(x,y)= a_0+ a_1x + a_2y \tag{1.6}$$

o, dicho de otra forma, se puede expresar a la variable de campo de la siguiente manera

$$T(x, y) = N_1(x, y)T_1 + N_2(x, y)T_2 + N_3(x, y)T_3 \tag{1.7}$$

donde N_1, N_2, N_3, son las funciones de interpolación que se buscan.

Se sabe que las temperaturas T_1, T_2, T_3, son las temperaturas correspondientes a los nodos del elemento. Entonces, sustituyendo las coordenadas de cada nodo del triángulo en la ecuación 1.6 se tiene lo siguiente:

$$T(x_1, y_1) = a_0 + a_1 x_1 + a_2 y_1 = T_1$$
$$T(x_2, y_2) = a_0 + a_1 x_2 + a_2 y_2 = T_2$$
$$T(x_3, y_3) = a_0 + a_1 x_3 + a_2 y_3 = T_3$$

Colocando en forma matricial el sistema de tres ecuaciones se tiene:

$$\begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \end{Bmatrix}$$

Para encontrar el valor de los coeficientes del sistema anterior se debe invertir la matriz 3x3, con lo que se llega a:

$$\begin{bmatrix} \frac{-x_2y_3+x_3y_2}{-x_1y_2+x_2y_1+x_1y_3-y_1x_3-x_2y_3+x_3y_2} & \frac{x_1y_3-x_3y_1}{-x_1y_2+x_2y_1+x_1y_3-y_1x_3-x_2y_3+x_3y_2} & \frac{-x_1y_2+x_2y_1}{-x_1y_2+x_2y_1+x_1y_3-y_1x_3-x_2y_3+x_3y_2} \\ \frac{-y_2+y_3}{-x_1y_2+x_2y_1+x_1y_3-y_1x_3-x_2y_3+x_3y_2} & \frac{y_1-y_3}{-x_1y_2+x_2y_1+x_1y_3-y_1x_3-x_2y_3+x_3y_2} & \frac{-y_1+y_2}{-x_1y_2+x_2y_1+x_1y_3-y_1x_3-x_2y_3+x_3y_2} \\ \frac{x_2-x_3}{-x_1y_2+x_2y_1+x_1y_3-y_1x_3-x_2y_3+x_3y_2} & \frac{-x_1+x_3}{-x_1y_2+x_2y_1+x_1y_3-y_1x_3-x_2y_3+x_3y_2} & \frac{x_1-x_2}{-x_1y_2+x_2y_1+x_1y_3-y_1x_3-x_2y_3+x_3y_2} \end{bmatrix}$$

de donde se obtiene que

$$a_o = \frac{1}{2A}\left[(x_2y_3 - x_3y_2)T_1 + (x_3y_1 - x_1y_3)T_2 + (x_1y_2 - x_2y_1)T_3\right]$$
$$a_1 = \frac{1}{2A}\left[(y_2 - y_3)T_1 + (y_3 - y_1)T_2 + (y_1 - y_2)T_3\right]$$
$$a_2 = \frac{1}{2A}\left[(x_3 - x_2)T_1 + (x_1 - x_3)T_2 + (x_2 - x_1)T_3\right]$$

Sustituyendo estos coeficientes en la ecuación 1.6 y agrupando los términos con respecto a las variables de campo se tiene lo siguiente

$$T(x,y) = \begin{Bmatrix} \left[(x_2y_3 - x_3y_2) + (y_2 - y_3)x + (x_3 - x_2)y\right]T_1 \\ +\left[(x_3y_1 - x_1y_3) + (y_3 - y_1)x + (x_1 - x_3)y\right]T_2 \\ +\left[(x_1y_2 - x_2y_1) + (y_1 - y_2)x + (x_2 - x_1)y\right]T_3 \end{Bmatrix}$$

de donde se obtienen las tres funciones de interpolación deseadas en términos de (x, y)

$$N_1(x,y)=\frac{1}{2A}\left[(x_2y_3-x_3y_2)+(y_2-y_3)x+(x_3-x_2)y\right] \quad (1.8)$$

$$N_2(x,y)=\frac{1}{2A}\left[(x_3y_1-x_1y_3)+(y_3-y_1)x+(x_1-x_3)y\right] \quad (1.9)$$

$$N_3(x,y)=\frac{1}{2A}\left[(x_1y_2-x_2y_1)+(y_1-y_2)x+(x_2-x_1)y\right] \quad (1.10)$$

A es el área del elemento triangular, que dada por las coordenadas de sus vértices es igual a

$$A=\frac{1}{2}\begin{vmatrix}1 & x_1 & y_1\\ 1 & x_2 & y_2\\ 1 & x_3 & y_3\end{vmatrix}$$

Una vez que se han obtenido las funciones de interpolación del elemento triangular se procede a obtener la matriz de rigidez de cada elemento, basándose para ello en la teoría de conducción bidimensional en estado estable junto con la teoría de Galerkin. Para esto se sabe que los problemas de conducción en dos dimensiones deben satisfacer la siguiente ecuación diferencial:

$$k_x t\frac{\partial^2 T}{\partial x^2}+k_y t\frac{\partial^2 T}{\partial y^2}=0 \quad (1.11)$$

Aplicando el método de Galerkin a la ecuación (1.11) se tienen las tres ecuaciones residuales que involucran a las funciones de interpolación anteriormente encontradas.

$$\int_A N_1\left(k_x t\frac{\partial^2 T}{\partial x^2}+k_y t\frac{\partial^2 T}{\partial y^2}\right)dA=0$$

$$\int_A N_2\left(k_x t\frac{\partial^2 T}{\partial x^2}+k_y t\frac{\partial^2 T}{\partial y^2}\right)dA=0$$

$$\int_A N_3\left(k_x t\frac{\partial^2 T}{\partial x^2}+k_y t\frac{\partial^2 T}{\partial y^2}\right)dA=0$$

Reescribiendo en forma matricial las tres ecuaciones anteriores se tiene:

$$\int_A [N]^T\left(k_x t\frac{\partial^2 T}{\partial x^2}+k_y t\frac{\partial^2 T}{\partial y^2}\right)dA=0$$

Separando la ecuación anterior en dos términos se tiene:

$$\int_A [N]^T k_x t\frac{\partial^2 T}{\partial x^2}dA+\int_A [N]^T k_y t\frac{\partial^2 T}{\partial y^2}dA=0 \quad (1.12)$$

Se sabe por las reglas de cálculo que la segunda derivada de una función lineal es igual a cero. Por lo tanto, si se utiliza las segundas derivadas de la temperatura siendo esta determinada por una función lineal, el resultado que se va a obtener va a ser igual a cero. Por esta razón se debe manipular estas segundas derivadas para obtener la ecuación en términos de derivadas de primer orden. Para esto se emplea la regla de la cadena.

$$\frac{\partial}{\partial x}\left(\left[N\right]^T \frac{\partial T}{\partial x}\right)=\left[N\right]^T \frac{\partial^2 T}{\partial x^2}+\frac{\partial\left[N\right]^T}{\partial x}\frac{\partial T}{\partial x}$$

Despejando el término con la segunda derivada se tiene:

$$\left[N\right]^T \frac{\partial^2 T}{\partial x^2}=\frac{\partial}{\partial x}\left(\left[N\right]^T \frac{\partial T}{\partial x}\right)-\left(\frac{\partial\left[N\right]^T}{\partial x}\frac{\partial T}{\partial x}\right)$$

Reemplazando la ecuación anterior en cada término de la ecuación 1.12 se tiene

$$\int_A \left[N\right]^T k_x t\frac{\partial^2 T}{\partial x^2}dA=\int_A k_x t\frac{\partial}{\partial x}\left(\left[N\right]^T \frac{\partial T}{\partial x}\right)dA-\int_A k_x t\left(\frac{\partial\left[N\right]^T}{\partial x}\frac{\partial T}{\partial x}\right)dA \qquad (1.13)$$

$$\int_A \left[N\right]^T k_y t\frac{\partial^2 T}{\partial y^2}dA=\int_A k_y t\frac{\partial}{\partial y}\left(\left[N\right]^T \frac{\partial T}{\partial y}\right)dA-\int_A k_y t\left(\frac{\partial\left[N\right]^T}{\partial y}\frac{\partial T}{\partial y}\right)dA \qquad (1.14)$$

Evaluando la segunda integral del lado derecho de la igualdad de la ecuación 1.13 se tiene

$$\frac{\partial\left[N\right]^T}{\partial x}=\frac{\partial}{\partial x}\begin{bmatrix}N_1\\N_2\\N_3\end{bmatrix}=\frac{1}{2A}\begin{bmatrix}y_2-y_3\\y_3-y_1\\y_1-y_2\end{bmatrix}$$

$$\frac{\partial T}{\partial x}=\frac{\partial}{\partial x}\begin{bmatrix}N_1 & N_2 & N_3\end{bmatrix}\begin{Bmatrix}T_1\\T_2\\T_3\end{Bmatrix}=\frac{1}{2A}\begin{bmatrix}y_2-y_3 & y_3-y_1 & y_1-y_2\end{bmatrix}\begin{Bmatrix}T_1\\T_2\\T_3\end{Bmatrix}$$

Sustituyendo estas derivadas en la integral se tiene

$$-\int_A k_x t\left(\frac{\partial\left[N\right]^T}{\partial x}\frac{\partial T}{\partial x}\right)dA=-\int_A k_x t\left(\frac{1}{4A^2}\begin{bmatrix}y_2-y_3\\y_3-y_1\\y_1-y_2\end{bmatrix}\begin{bmatrix}y_2-y_3 & y_3-y_1 & y_1-y_2\end{bmatrix}\begin{Bmatrix}T_1\\T_2\\T_3\end{Bmatrix}\right)dA$$

Integrando en el área resulta

$$\frac{k_x t}{4A}\begin{bmatrix} (y_2-y_3)^2 & (y_2-y_3)(y_3-y_1) & (y_2-y_3)(y_1-y_2) \\ (y_3-y_1)(y_2-y_3) & (y_3-y_1)^2 & (y_3-y_1)(y_1-y_2) \\ (y_1-y_2)(y_2-y_3) & (y_1-y_2)(y_3-y_1) & (y_1-y_2)^2 \end{bmatrix}\begin{Bmatrix} T_1 \\ T_2 \\ T_3 \end{Bmatrix}$$

Se procede de la misma manera con la segunda integral del lado derecho de la igualdad de la ecuación 1.14, obteniéndose el siguiente resultado:

$$\frac{k_y t}{4A}\begin{bmatrix} (x_3-x_2)^2 & (x_3-x_2)(x_1-x_3) & (x_3-x_2)(x_2-x_1) \\ (x_1-x_3)(x_3-x_2) & (x_1-x_3)^2 & (x_1-x_3)(x_2-x_1) \\ (x_2-x_1)(x_3-x_2) & (x_2-x_1)(x_1-x_3) & (x_2-x_1)^2 \end{bmatrix}\begin{Bmatrix} T_1 \\ T_2 \\ T_3 \end{Bmatrix}$$

Finalmente, la matriz de rigidez para un elemento triangular bajo condiciones de conducción bidimensional en estado estable es igual a:

$$[K]^e = \frac{k_x t}{4A}\begin{bmatrix} (y_2-y_3)^2 & (y_2-y_3)(y_3-y_1) & (y_2-y_3)(y_1-y_2) \\ (y_3-y_1)(y_2-y_3) & (y_3-y_1)^2 & (y_3-y_1)(y_1-y_2) \\ (y_1-y_2)(y_2-y_3) & (y_1-y_2)(y_3-y_1) & (y_1-y_2)^2 \end{bmatrix} + \frac{k_y t}{4A}\begin{bmatrix} (x_3-x_2)^2 & (x_3-x_2)(x_1-x_3) & (x_3-x_2)(x_2-x_1) \\ (x_1-x_3)(x_3-x_2) & (x_1-x_3)^2 & (x_1-x_3)(x_2-x_1) \\ (x_2-x_1)(x_3-x_2) & (x_2-x_1)(x_1-x_3) & (x_2-x_1)^2 \end{bmatrix}$$

A la expresión anterior se le debe sumar los siguientes términos si es que la naturaleza del problema involucra el fenómeno de convección.

$$[K]^e = \frac{htl_{ij}}{6}\begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad [K]^e = \frac{htl_{jk}}{6}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix} \qquad [K]^e = \frac{htl_{ki}}{6}\begin{bmatrix} 2 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 2 \end{bmatrix}$$

Los subíndices *i, j, k* representan la posición de los nodos de cada elemento triangular, siendo *l* la longitud del lado del elemento triangular expuesto al fenómeno de convección.

Por otra parte, los vectores de fuerzas pueden estar compuestos por un término dado por generación interna dentro del dominio o del elemento, un término dado por conducción en la superficie del elemento, y un tercer término dado por convección.

$$\{F\}^e = \frac{GAt}{3}\begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} - \frac{qtl_{ij}}{2}\begin{Bmatrix} 1 \\ 1 \\ 0 \end{Bmatrix} + \frac{hT_a tl_{jk}}{2}\begin{Bmatrix} 0 \\ 1 \\ 1 \end{Bmatrix}$$

La posición de los valores entre llaves varía de acuerdo a la posición de los nodos del elemento, pudiendo hacerse combinaciones *ij*, *jk*, *kj* y *ki*.

1.5.2. Elementos rectangulares

La variable de campo de los problemas de transferencia de calor utilizando elementos rectangulares se puede expresar de la siguiente manera

$$T(x, y) = b_0 + b_1 x + b_2 y + b_3 xy \tag{1.15}$$

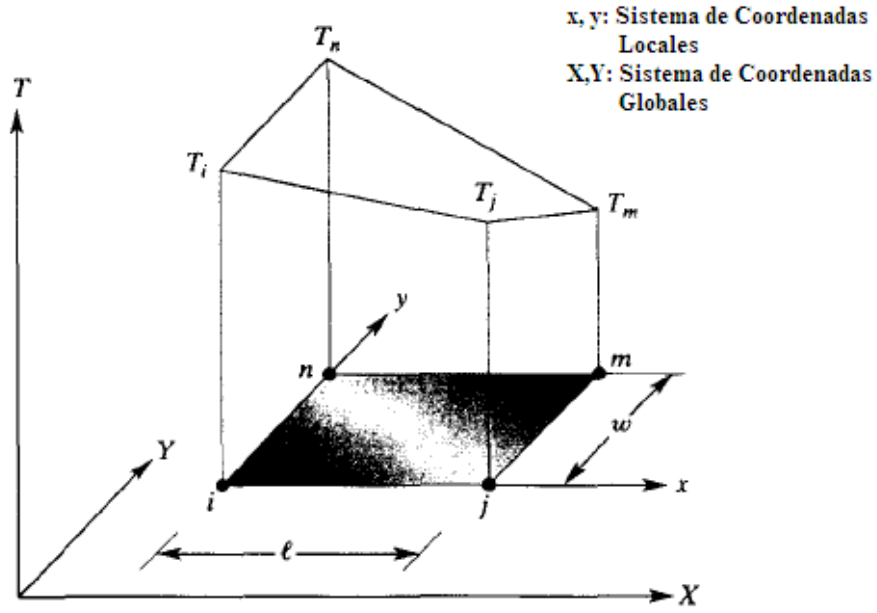


Figura 1.8. Elemento rectangular. [8]

Se conoce que las temperaturas T_1, T_2, T_3, T_4 son las temperaturas correspondientes a los nodos del elemento. Entonces, sustituyendo las coordenadas de cada nodo del rectángulo en la ecuación 1.15 se tiene lo siguiente:

$$T(x_1, y_1) = b_0 + b_1 x_1 + b_2 y_1 + b_3 x_1 y_1 = T_1$$
$$T(x_2, y_2) = b_0 + b_1 x_2 + b_2 y_2 + b_3 x_2 y_2 = T_2$$
$$T(x_3, y_3) = b_0 + b_1 x_3 + b_2 y_3 + b_3 x_3 y_3 = T_3$$
$$T(x_4, y_4) = b_0 + b_1 x_4 + b_2 y_4 + b_3 x_4 y_4 = T_4$$

Colocando en forma matricial el sistema de cuatro ecuaciones se tiene:

$$\begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 \\ 1 & x_2 & y_2 & x_2 y_2 \\ 1 & x_3 & y_3 & x_3 y_3 \\ 1 & x_4 & y_4 & x_4 y_4 \end{bmatrix} \begin{Bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{Bmatrix} = \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{Bmatrix}$$

Invirtiendo la matriz 4x4 se encuentran los valores de los coeficientes, pero por resultar en expresiones muy complejas se utilizan las coordenadas locales del elemento, con lo que se debe cumplir las siguientes condiciones en las temperaturas nodales.

$$T = T_i \quad en \quad x = 0, y = 0$$
$$T = T_j \quad en \quad x = l,\ y = 0$$
$$T = T_m \quad en \quad x = l,\ y = w$$
$$T = T_n \quad en \quad x = 0, y = w$$

Aplicando las condiciones expresadas anteriormente y reagrupando los términos en función de las temperaturas se obtienen las siguientes funciones de interpolación:

$$N_1(x,y) = \left(1 - \frac{x}{l}\right)\left(1 - \frac{y}{w}\right)$$
$$N_2(x,y) = \frac{x}{l}\left(1 - \frac{y}{w}\right)$$
$$N_3(x,y) = \frac{xy}{lw}$$
$$N_4(x,y) = \frac{y}{w}\left(1 - \frac{x}{l}\right)$$

Una vez obtenidas las funciones de interpolación se prosigue a obtener la matriz de rigidez del elemento rectangular siguiendo los mismos pasos que se realizaron para la obtención de la matriz de rigidez del elemento triangular. Primero se aplica el método de Galerkin y se obtiene una ecuación para los residuales de las cuatro funciones de interpolación

$$\int_A [N]^T \left(k_x t \frac{\partial^2 T}{\partial x^2} + k_y t \frac{\partial^2 T}{\partial y^2} \right) dA = 0$$

Separando la ecuación anterior en dos términos se tiene:

$$\int_A [N]^T k_x t \frac{\partial^2 T}{\partial x^2} dA + \int_A [N]^T k_y t \frac{\partial^2 T}{\partial y^2} dA = 0 \qquad (1.16)$$

Utilizando la regla de la cadena se manipula la segunda derivada para dejar todo en función de las primeras derivadas.

$$\frac{\partial}{\partial x}\left([N]^T \frac{\partial T}{\partial x} \right) = [N]^T \frac{\partial^2 T}{\partial x^2} + \frac{\partial [N]^T}{\partial x} \frac{\partial T}{\partial x}$$

Despejando el término con la segunda derivada se tiene:

$$[N]^T \frac{\partial^2 T}{\partial x^2} = \frac{\partial}{\partial x}\left([N]^T \frac{\partial T}{\partial x} \right) - \left(\frac{\partial [N]^T}{\partial x} \frac{\partial T}{\partial x} \right)$$

Reemplazando la ecuación anterior en la ecuación 1.16 se tiene:

$$\int_A [N]^T k_x t \frac{\partial^2 T}{\partial x^2} dA = \int_A k_x t \frac{\partial}{\partial x}\left([N]^T \frac{\partial T}{\partial x}\right) dA - \int_A k_x t \left(\frac{\partial [N]^T}{\partial x}\frac{\partial T}{\partial x}\right) dA \qquad (1.17)$$

$$\int_A [N]^T k_y t \frac{\partial^2 T}{\partial y^2} dA = \int_A k_y t \frac{\partial}{\partial y}\left([N]^T \frac{\partial T}{\partial y}\right) dA - \int_A k_y t \left(\frac{\partial [N]^T}{\partial y}\frac{\partial T}{\partial y}\right) dA \qquad (1.18)$$

Evaluando la segunda integral del lado derecho de la igualdad de la ecuación 1.17 se tiene:

$$\frac{\partial [N]^T}{\partial x} = \frac{\partial}{\partial x}\begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{bmatrix} = \frac{1}{lw}\begin{bmatrix} -w+y \\ w-y \\ y \\ -y \end{bmatrix}$$

$$\frac{\partial T}{\partial x} = \frac{\partial}{\partial x}\begin{bmatrix} N_1 & N_2 & N_3 & N_4 \end{bmatrix}\begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{Bmatrix} = \frac{1}{lw}\begin{bmatrix} (-w+y) & (w-y) & y & -y \end{bmatrix}\begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{Bmatrix}$$

Sustituyendo estas derivadas en la integral se tiene:

$$-\int_A k_x t \left(\frac{\partial [N]^T}{\partial x}\frac{\partial T}{\partial x}\right) dA = -\int_A k_x t \left(\frac{1}{(lw)^2}\begin{bmatrix} -w+y \\ w-y \\ y \\ -y \end{bmatrix}\begin{bmatrix} (-w+y) & (w-y) & y & -y \end{bmatrix}\begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{Bmatrix}\right) dA$$

Integrando en el área se tiene

$$-\frac{k_x tw}{6l}\begin{bmatrix} 2 & -2 & -1 & 1 \\ -2 & 2 & 1 & -1 \\ -1 & 1 & 2 & -2 \\ 1 & -1 & -2 & 2 \end{bmatrix}\begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{Bmatrix}$$

Se procede de la misma manera con la segunda integral del lado derecho de la igualdad de la ecuación 1.18, obteniéndose el siguiente resultado:

$$-\frac{k_y tl}{6w}\begin{bmatrix} 2 & 1 & -1 & -2 \\ 1 & 2 & -2 & -1 \\ -1 & -2 & 2 & 1 \\ -2 & -1 & 1 & 2 \end{bmatrix}\begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{Bmatrix}$$

Finalmente, la matriz de rigidez para un elemento rectangular bajo condiciones de conducción bidimensional en estado estable es igual a:

$$[K]^e = \frac{k_x tw}{6l}\begin{bmatrix} 2 & -2 & -1 & 1 \\ -2 & 2 & 1 & -1 \\ -1 & 1 & 2 & -2 \\ 1 & -1 & -2 & 2 \end{bmatrix} + \frac{k_y tl}{6w}\begin{bmatrix} 2 & 1 & -1 & -2 \\ 1 & 2 & -2 & -1 \\ -1 & -2 & 2 & 1 \\ -2 & -1 & 1 & 2 \end{bmatrix}$$

De igual manera que con el elemento triangular, a esta expresión de la matriz de rigidez se debe sumar otros componentes debidos al fenómeno de convección.

$$[K]^e = \frac{htl_{ij}}{6}\begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad [K]^e = \frac{htl_{jm}}{6}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$[K]^e = \frac{htl_{mn}}{6}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix} \qquad [K]^e = \frac{htl_{ni}}{6}\begin{bmatrix} 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 \end{bmatrix}$$

De manera similar, el vector de fuerzas va a ser el siguiente:

$$\{F\}^e = \frac{QAt}{4}\begin{Bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{Bmatrix} - \frac{qtl_{ij}}{2}\begin{Bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{Bmatrix} + \frac{hT_a tl_{jm}}{2}\begin{Bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{Bmatrix}$$

La posición de los valores entre llaves varía de acuerdo a la posición de los nodos del elemento, pudiendo hacerse combinaciones *ij*, *jm*, *mn*, *ni*. Estas combinaciones se las realiza dependiendo en donde se aplica la carga, sea este, un flujo de calor o un flujo de convección, entonces analizando cada elemento y verificando en cuales de los nodos se encuentra presente una carga se arma el vector colocando el valor de 1 en las posiciones de los nodos con carga.

Cuando se han obtenido las matrices de rigidez para cada elemento, sea este triangular o rectangular, se prosigue con el ensamblaje del sistema global de ecuaciones y la posterior resolución del mismo. Así se pueden obtener los valores de temperaturas y los valores de fuerzas para cada nodo.

1.6. Ejemplo de cálculo

A continuación se va desarrollar un ejemplo de cálculo, ejemplo que se lo va a resolver mediante el método analítico y mediante el método de los elementos finitos para comprobar que los resultados obtenidos por ambos métodos son iguales o similares.

1.6.1. Flujo de calor a través de una placa metálica

Se tiene una placa metálica de espesor unitario con una conductividad térmica (k = 10 W/mK) con un largo de 1 m y un ancho de 1 m. Las superficies laterales y la superficie inferior se encuentran a una temperatura de 100K, mientras que la superficie superior tiene una temperatura aproximada de 500K. Se busca determinar la temperatura en la posición (0.5 m, 0.5 m), además del flujo de calor que atraviesa la placa.

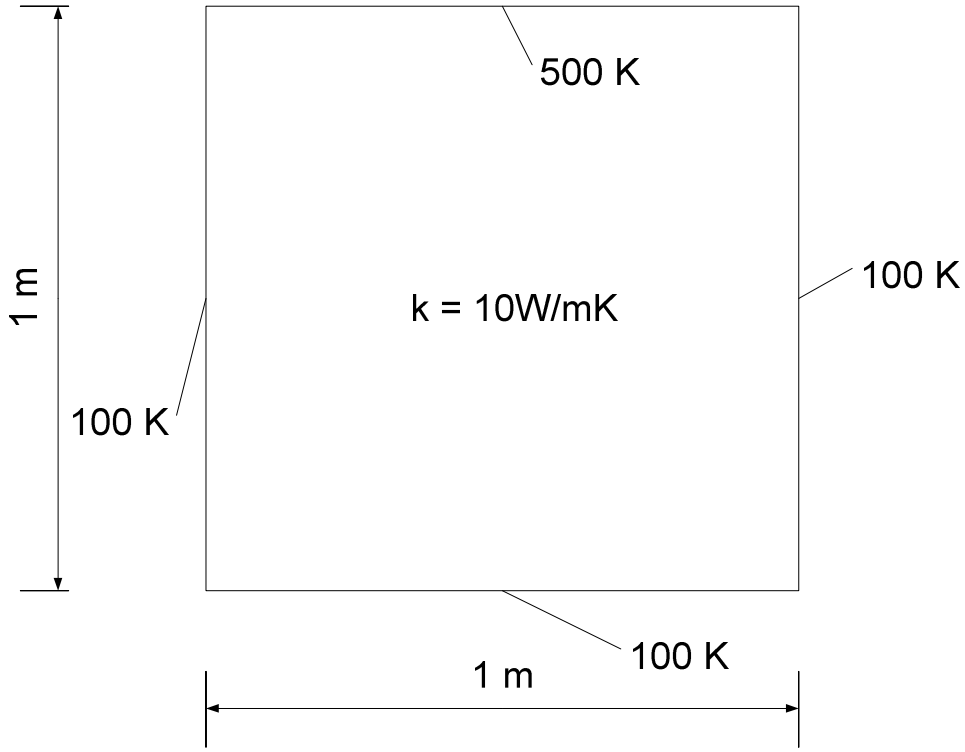


Figura 1.9. Geometría del ejemplo de cálculo 1.

Método Analítico

Por ser un problema de conducción en dos dimensiones y dado que la conductividad térmica es constante, la ecuación diferencial que rige este problema es la siguiente:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

Las condiciones de frontera del problema son:

$$\begin{aligned} &x = 0, \quad 0 < y < 1; \quad T = 100 \\ &y = 0, \quad 0 < x < 1; \quad T = 100 \\ &x = 1, \quad 0 < y < 1; \quad T = 100 \\ &y = 1, \quad 0 < x < 1; \quad T = 500 \end{aligned}$$

Con la ecuación diferencial del problema y las condiciones de frontera anteriormente especificadas se procede a resolver el problema analíticamente. Para ello se utiliza el método de separación de variables. Se supone que la función de temperatura T(x,y) es de la forma:

$$T(x, y) = X(x)Y(y)$$

Reemplazando la ecuación anterior en la ecuación diferencial del problema se obtiene una expresión en la cual las variables *x* y *y* están separadas y son independientes una de la otra. La expresión obtenida es la siguiente:

$$-\frac{1}{X}\frac{d^2X}{dx^2}=\frac{1}{Y}\frac{d^2Y}{dy^2}$$

Igualando cada lado de la expresión anterior a una constante (β) se obtiene dos ecuaciones diferenciales ordinarias:

$$\frac{d^2X}{dx^2}+\beta X=0$$
$$\frac{d^2Y}{dy^2}-\beta Y=0$$

Una vez propuestas estas dos ecuaciones diferenciales ordinarias, se obtiene la solución no trivial para cada una de ellas, obteniéndose finalmente con ayuda de Series de Fourier que la distribución de temperaturas para este problema viene dada por:

$$T(x,y)=(Ts-To)\left[\frac{2}{\pi}\sum_{n=1}^{\infty}\frac{(-1)^{n+1}+1}{n}\times sen\frac{n\pi x}{a}\frac{senh\left(\frac{n\pi y}{a}\right)}{senh\left(\frac{n\pi b}{a}\right)}\right]+To$$

De donde *Ts* es la temperatura de la superficie superior, *To* representa la temperatura de las superficies laterales e inferior, *a* es el ancho de la placa y *b* es la altura de la placa, reemplazando los valores dados en la ecuación de la distribución de temperaturas, se obtiene el valor de la temperatura para la posición x = 0.5 m, y = 0.5 m.

$$\mathrm{T}(0.5,0.5)=\left((500-100)\frac{2}{\pi}\sum_{\mathrm{n}=1}^{\infty}\frac{(-1)^{\mathrm{n}+1}+1}{\mathrm{n}}\mathrm{sen}\frac{\mathrm{n}\pi(0.5)}{1}\frac{\mathrm{senh}\left(\frac{\mathrm{n}\pi(0.5)}{1}\right)}{\mathrm{senh}\left(\frac{\mathrm{n}\pi(1)}{(1)}\right)}\right)+100=200\mathrm{K}$$

La tasa de flujo de calor para la posición entre la mitad de la placa y la superficie inferior se obtiene usando la ley de Fourier:

$$\mathrm{q_y}=\left(\frac{\mathrm{k}\Delta\mathrm{T}}{\mathrm{L}}\right)(\mathrm{A})=\left(10\frac{200-100}{1}\right)1=1000\,\mathrm{W}$$

Método de los Elementos Finitos

Para usar el método de los elementos finitos, primero se selecciona un tipo de elemento para discretizar la geometría del problema. Para este ejemplo se va a utilizar elementos triangulares y se va a dividir el dominio del problema en 8 elementos. La figura 1.10 muestra la geometría ya segmentada.

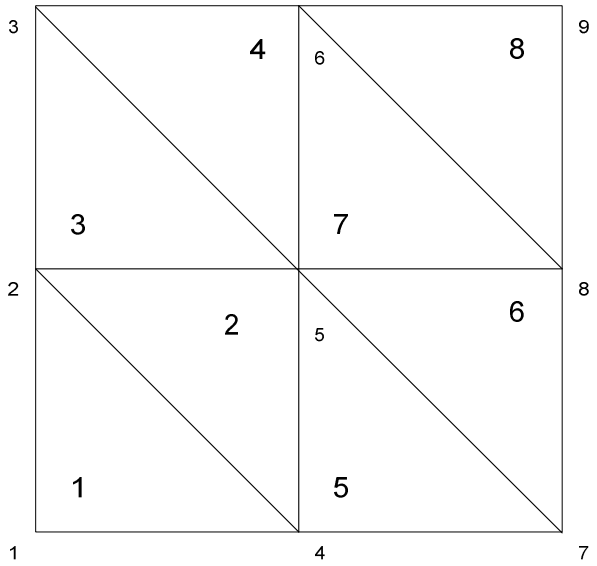


Figura 1.10. Geometría del ejemplo de cálculo 1 mallada en 8 elementos.

Una vez mallada la geometría e identificados todos sus nodos y elementos, el siguiente paso es encontrar la matriz de rigidez para cada elemento. En este caso los elementos 1, 3, 5 y 7 tienen la misma matriz de rigidez mientras que los elementos 2, 4, 6 y 8 tienen una distinta.

De la sección anterior se obtuvo la matriz de rigidez para los elementos triangulares:

$$[k]^e = \frac{k_x t}{4A}\begin{bmatrix} (y_2-y_3)^2 & (y_2-y_3)(y_3-y_1) & (y_2-y_3)(y_1-y_2) \\ (y_3-y_1)(y_2-y_3) & (y_3-y_1)^2 & (y_3-y_1)(y_1-y_2) \\ (y_1-y_2)(y_2-y_3) & (y_1-y_2)(y_3-y_1) & (y_1-y_2)^2 \end{bmatrix} + \frac{k_y t}{4A}\begin{bmatrix} (x_3-x_2)^2 & (x_3-x_2)(x_1-x_3) & (x_3-x_2)(x_2-x_1) \\ (x_1-x_3)(x_3-x_2) & (x_1-x_3)^2 & (x_1-x_3)(x_2-x_1) \\ (x_2-x_1)(x_3-x_2) & (x_2-x_1)(x_1-x_3) & (x_2-x_1)^2 \end{bmatrix}$$

Entonces, para los elementos 1, 3, 5 y 7 se tiene la siguiente matriz de rigidez:

$$[k]^1 = \frac{10}{2}\begin{bmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 10 & -5 & -5 \\ -5 & 5 & 0 \\ -5 & 0 & 5 \end{bmatrix} \text{[W/K]}$$

Para los elementos 2, 4, 6 y 8 la matriz de rigidez es la siguiente:

$$[k]^2 = \frac{10}{2}\begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 5 & -5 & 0 \\ -5 & 10 & 5 \\ 0 & -5 & 5 \end{bmatrix} \text{[W/K]}$$

Una vez obtenida las matrices de rigidez de todos los elementos, el siguiente paso es ensamblar la matriz de rigidez global en base a la conectividad de los nodos con cada elemento. Al final del ensamblaje resulta una matriz de orden 9x9:

$$Ksist = \begin{bmatrix} 10 & -5 & 0 & -5 & 0 & 0 & 0 & 0 & 0 \\ -5 & 20 & -5 & 0 & -10 & 0 & 0 & 0 & 0 \\ 0 & -5 & 10 & 0 & 0 & -5 & 0 & 0 & 0 \\ -5 & 0 & 0 & 20 & -10 & 0 & -5 & 0 & 0 \\ 0 & -10 & 0 & -10 & 40 & -10 & 0 & -10 & 0 \\ 0 & 0 & -5 & 0 & -10 & 20 & 0 & 0 & -5 \\ 0 & 0 & 0 & -5 & 0 & 0 & 10 & -5 & 0 \\ 0 & 0 & 0 & 0 & -10 & 0 & -5 & 20 & -5 \\ 0 & 0 & 0 & 0 & 0 & -5 & 0 & -5 & 10 \end{bmatrix} [W/K]$$

Cuando se ha ensamblado la matriz y con las condiciones de frontera se construye la ecuación [K]{T}={Q}, donde el vector {T} es el vector de las temperaturas que se desea obtener. El vector de temperaturas es el siguiente:

$$\{T\} = [K]^{-1}\{Q\} = \{T\} = \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \\ T_9 \end{Bmatrix} = \begin{Bmatrix} 100 \\ 100 \\ 500 \\ 100 \\ 200 \\ 500 \\ 100 \\ 100 \\ 500 \end{Bmatrix} [K]$$

Comparando las respuestas obtenidas por el método analítico y por el MEF, se ve que en ambos casos la temperatura calculada es la misma. Una vez calculadas las temperaturas y obtenida la matriz de rigidez se calcula el vector {Q} de los flujos en [Watts]. Este vector muestra los valores del flujo en Watts en cada uno de los nodos de la placa.

$$\{Q\} = [K]\ \{T\} = \{Q\} = \begin{Bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \\ Q_5 \\ Q_6 \\ Q_7 \\ Q_8 \\ Q_9 \end{Bmatrix} = \begin{Bmatrix} 0 \\ -3000 \\ 2000 \\ -1000 \\ 0 \\ 3000 \\ 0 \\ -3000 \\ 2000 \end{Bmatrix}$$

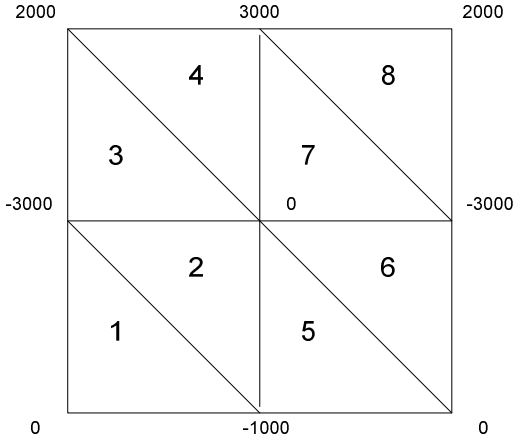


Figura 1.11. Flujos resultantes en geometría mallada en 8 elementos.

Como se puede apreciar según la distribución de los nodos, los valores de flujo de calor entre la mitad de la placa y la superficie inferior se obtiene con la resta del valor Q_5 menos el valor de Q_4. Dando como resultado 1000 [W], igual resultado que la solución analítica.

CAPÍTULO 2

EL LENGUAJE DE PROGRAMACIÓN JAVA

En este capítulo se tratan brevemente algunos tópicos relacionados con el lenguaje de programación Java. Su origen y evolución como lenguaje de programación, sus características principales así como las ventajas y desventajas en comparación con otros lenguajes. También se describen algunos de los elementos que en el mercado existen para desarrollar aplicaciones con Java y se brinda una breve explicación sobre su nomenclatura y la estructuración de una aplicación.

2.1. Historia del lenguaje de programación Java

En un principio el lenguaje de programación Java estaba destinado para la programación de electrodomésticos, y debido a la poca capacidad de almacenamiento de estos, la idea de Java era la de ser un lenguaje sencillo con generación de código también sencilla. Pero debido a que cada electrodoméstico poseía una unidad controladora de procesos (CPU) el lenguaje creado debía funcionar para cualquier electrodoméstico. Por ello Java se desarrolló bajo un código "neutro", logrando así que el lenguaje pueda ser usado por cualquier CPU de cualquier electrodoméstico. Lastimosamente Java no prosperó como un lenguaje utilizado por los fabricantes de electrodomésticos.

Tres de las principales razones que llevaron a crear Java son:

1. Creciente necesidad de interfaces mucho más cómodas e intuitivas que los sistemas de ventanas que proliferaban hasta el momento.
2. Fiabilidad del código y facilidad de desarrollo. El creador de Java, Gosling, observó que muchas de las características que ofrecían C o C++ aumentaban de forma alarmante el gran coste de pruebas y depuración. Por ello, en sus ratos libres creó un lenguaje de programación donde intentaba solucionar los fallos que encontraba en C++.
3. Enorme diversidad de controladores electrónicos. Los dispositivos electrónicos se controlan mediante la utilización de microprocesadores de bajo precio y reducidas prestaciones, que varían cada poco tiempo y que utilizan diversos conjuntos de instrucciones. Java permite escribir un código común para todos los dispositivos. [16]

El enfoque de Java cambió drásticamente debido a la aparición del Internet, con lo cual los creadores del lenguaje vieron una gran oportunidad para el desarrollo y auge de Java. Por esta razón se empezaron a crear programas, compiladores, los llamados JDK Java Development Kit con los cuales se permitía a los usuarios desarrollar aplicaciones en Java llenas de gráficos, animaciones y sonidos.

Actualmente Java es un lenguaje de programación ampliamente usado, y no solo para aplicaciones que involucren páginas web o Internet. También se están desarrollando aplicaciones y programas basados en Java pero con otro enfoque, como puede ser programas de cálculo numérico, juegos, programas para soluciones estratégicas en los negocios, entre otras aplicaciones que permite codificar el lenguaje.

2.2. Características del lenguaje Java

Java es un lenguaje de programación interpretado y de alto nivel, con ciertas características que lo hacen un lenguaje de programación muy usado. El lenguaje Java es: simple, orientado a objetos, distribuido, multitarea, dinámico, de alto desempeño, robusto y seguro.

En realidad, Java es un lenguaje muy parecido a C++, pero en Java ciertas características y ciertos métodos complejos de C++ se eliminaron. Este es el caso de los punteros, la herencia múltiple, etc. El programa conserva algunas características básicas como la sintaxis, la programación orientada a objetos y el uso de clases.

El lenguaje Java permite la interacción con Internet, posee una API (Application Programming Interface), que es una interfaz para el manejo de gráficos y archivos multimedia, así como también librerías como la JDBC que permiten interactuar con otras bases de datos externas a la aplicación.

Para llegar a un programa final de Java, este debe pasar por algunos pasos y procesos. Primero el archivo con extensión .java es compilado en un código conocido como bytecode y se lo organiza en archivos clases de extensión .class. Posteriormente estas clases son interpretadas por una máquina virtual, la cual utiliza tanto el sistema operativo como el hardware de la máquina subyacente. “Tanto es así, que una máquina virtual Java consta de los mismos elementos que un procesador: un juego de instrucciones, un conjunto de registros, una pila, además de un recolector de basura y un área de métodos. Además, la máquina permite la ejecución de varios hilos (multithreaded) de forma simultánea. La diferencia fundamental es que, al no ser un verdadero microprocesador, el bytecode se interpreta y no se ejecutan las instrucciones del computador directamente. La estructura de esta máquina proporciona dos cualidades muy importantes: portabilidad y seguridad.” [13]

2.2.1. Ventajas y desventajas de Java sobre otros lenguajes de programación

Como se dijo anteriormente, algunas de las características de Java son semejantes a las de C y C++, pero Java presenta ciertas ventajas sobre estos dos lenguajes. Una de las principales ventajas de Java es que al eliminar ciertas funciones propias de C y C++ se reduce en un 50% los errores más comunes al momento de programar, haciendo de Java un lenguaje más simple de usar y con menor probabilidad de tener errores de programación.

Otra ventaja de Java sobre otros lenguajes de programación es que Java es un lenguaje extremadamente seguro, ya que a diferencia de C y C++, Java elimina los *punteros*, lo que permite que no se tenga acceso a cualquier sector de la memoria y que no se pueda así extraer información confidencial de la máquina. Por otra parte, Java realiza una serie de pruebas antes de ejecutarse, evitando así fragmentos de código que sean ilegales o que realicen operaciones no deseadas.

Una de las desventajas que presenta Java en relación a C++ es la lentitud del primero frente al segundo. Esto se debe a que Java es un lenguaje de arquitectura neutra, es decir, un lenguaje que funciona independientemente del sistema operativo de la máquina. Por ello no existen compiladores específicos para cada sistema operativo por lo que la aplicación debe ser interpretada y no ejecutada como en el caso de otros lenguajes de programación.

Finalmente, una de las principales ventajas de Java sobre los demás lenguajes de programación es que Java es un lenguaje de distribución gratuita. No se necesita invertir grandes sumas de dinero en licencias para tener acceso a los desarrolladores, compiladores y librerías utilizadas por Java para la creación de un sinnúmero de aplicaciones.

2.3. Elementos de desarrollo para aplicaciones en Java

En el mercado existen programas que permiten a los usuarios crear y desarrollar programas bajo el lenguaje Java. La mayoría de estos programas incluyen opciones que le permiten al usuario codificar, compilar y correr la aplicación sin la necesidad de software adicional.

Existen los llamados ***Java Development Kit*** **(JDK)**, que contienen una serie de programas, librerías que permiten codificar, compilar y ejecutar aplicaciones en lenguaje Java. También están los ***Java Runtime Environment*** **(JRE)**, los cuales solo permiten ejecutar las aplicaciones más no desarrollarlas ni compilarlas. También están los ***Integrated Development Environment*** **(IDE)**. En este caso, en un mismo programa es posible escribir el código *Java*, compilarlo y ejecutarlo sin tener que cambiar de aplicación. Estos entornos integrados permiten desarrollar las aplicaciones de forma mucho más rápida. Para desarrollar la aplicación Placa se escogió este tipo de herramienta, por permitir realizar todas las tareas necesarias en un solo programa. En particular, se escogió NetBeans porque al ser un IDE desarrollado por Sun, es de distribución gratuita y fue el entorno en el que se desarrolló la primera versión del programa Placa, por lo que resultaba más rápido y menos complejo desarrollar la segunda versión utilizando esta herramienta. A parte de NetBeans, existen en el mercado algunos otros IDEs que se utilizan en el desarrollo de aplicaciones en java. La mayoría de estos IDEs, al igual que NetBeans, son funcionales en varios sistemas operativos, como Windows, Linux o Mac OS. Algunos ejemplos de estos son: *BlueJ, CodeGuide, Eclipse JDT, JDeveloper, IntelliJIDEA,* entre otros.

A continuación se explicará más detalladamente cada una de las herramientas de desarrolladores que permiten crear un programa en Java.

2.3.1. El compilador de Java

El compilador de Java es una de las herramientas presentes en los programas de desarrollo explicados anteriormente, sea en el JDK o en el IDE. La función que tiene el compilador es analizar el código escrito de los archivos *.java en busca de errores de sintaxis. Si no se encuentran errores, el compilador genera los ficheros con extensión *.class; caso contrario el compilador muestra los lugares en donde se han producido errores.

2.3.2. La Java Virtual Machine

Como se mencionó al principio del capítulo, el lenguaje Java en un principio estaba destinado a la programación de electrodomésticos, pero como cada electrodoméstico poseía su propio tipo de procesador y de ordenador, los creadores de Java se vieron obligados a desarrollar un software que no dependiera de los tipos de procesadores. Por tal razón desarrollaron un código neutro el cual debía ser ejecutado sobre una máquina virtual a la cual se la llamó ***Java Virtual Machine*** **(JVM)**. La función de esta máquina virtual es la de interpretar el código neutro para generar un código propio del procesador que se está utilizando, evitando así la pérdida de tiempo al tener que desarrollar un código diferente para cada tipo de procesador u ordenador.

"La *JVM* es el intérprete de *Java*. Ejecuta los "*bytecodes*" (ficheros compilados con extensión ****.class***) creados por el compilador de *Java* (*javac.exe*). Tiene numerosas opciones entre las que destaca la posibilidad de utilizar el denominado ***JIT*** (*Just-In-Time Compiler*), que puede mejorar entre 10 y 20 veces la velocidad de ejecución de un programa." [13]

2.4. Nomenclatura utilizada en el lenguaje Java

Durante la programación en Java es muy común que se deban usar varios nombres para especificar distintas variables del programa. Para Java existe una diferencia en lo que se refiere a letras mayúsculas y minúsculas. Así, pueden existir variables con el mismo nombre, como por ejemplo: Flujos, flujos y FLUJOS; y aunque uno pensaría que se refieren a la misma variable, Java considera que las tres son distintas variables. Por esta razón se debe tener mucho cuidado al momento de escribir un código para una aplicación en Java, se deberían seguir normas que faciliten el seguimiento del programa, así como su mantenimiento y lectura. Las siguientes son algunas recomendaciones sobre como se debería utilizar la nomenclatura al momento de codificar en Java:

1. En *Java* es habitual utilizar nombres con minúsculas, con las excepciones que se indican en los puntos siguientes.
2. Cuando un nombre consta de varias palabras es habitual poner una a continuación de otra, escribiendo con mayúscula la primera letra de la palabra que sigue a otra (Ejemplos: *VentanaCerrable*, *RectanguloGrafico*, *addWindowListener()*).
3. Los nombres de ***clases*** e ***interfaces*** comienzan siempre por mayúscula (Ejemplos: *Geometria*, *Rectangulo*, *Dibujable*, *Graphics*, *ArrayList*, *Iterator*).
4. Los nombres de ***objetos***, los nombres de ***métodos*** y ***variables miembro***, y los nombres de las ***variables locales*** de los métodos, comienzan siempre con minúscula (Ejemplos: *main()*, *dibujar()*, *numRectangulos*, *x*, *y*, *r*).
5. Los nombres de las ***variables finales***, es decir de las constantes, se definen siempre con mayúsculas (Ejemplo: *PI*). [13]

2.5. Estructuración de una aplicación en Java

En los programas desarrollados en lenguaje Java se debe tener siempre una clase importante la cual contiene la aplicación principal. Esta aplicación principal contiene dentro de su código la clase *main()* así como otras clases auxiliares creadas por el usuario que son de utilidad para el desarrollo del programa. Los archivos fuente utilizados por el programa siempre tienen la extensión *.java* y al momento de compilar estos archivos se crean los archivos *.class*

Un archivo fuente (**.java*) debe contener una sola clase de índole *public*, y esta clase debe llevar el mismo nombre del archivo fuente. Es decir, si el archivo fuente es *Unidades.java*, la declaración de la clase en el archivo debe ser *public class Unidades().* Además de la clase *public*, los archivos fuente pueden contener otras clases las cuales no necesariamente tienen que coincidir con el nombre el archivo *.java*. Como para toda codificación en Java, se tiene que tomar en cuenta que los nombres coincidan en la forma de escritura ya que pueden surgir errores al momento de compilar si se confunden mayúsculas con minúsculas.

2.5.1. Clase

Una clase es un conjunto de información que se divide en ***datos*** (variables o campos) y en ***funciones*** (métodos) que operan sobre esos datos. La programación en Java por ser orientada a objetos se basa principalmente en la programación de estas clases. Los programas en Java se construyen a partir de un conjunto de varias clases.

Cuando ya se definió e implementó una clase, se puede crear o declarar elementos propios de una clase denominados ***objetos*** de la clase. Estos objetos se pueden declarar al igual que se declaran las variables (de los tipos primitivos *int*, *double*, *String*,…). Cada objeto de la clase posee una copia de las variables que pertenecen a la clase del mismo objeto. Estas copias poseen sus propios valores que difieren de los demás objetos de la clase.

2.5.2. Herencia

La herencia es un concepto y una herramienta muy útil al momento de codificar un programa en Java. La herencia tiene como utilidad permitir que nuevas clases hereden las características de clases existentes. Es decir, que las nuevas clases se basan en las ya existentes y no se debe volver a codificar nada para la nueva clase pues al heredar las características de las clases existentes se reutiliza el código ya escrito. La nueva clase además de heredar todas las variables y métodos, también puede contener variables propias de esa clase así como métodos nuevos.

"En *Java*, a diferencia de otros lenguajes orientados a objetos, una clase solo se puede derivar de una única clase, con lo cual no es posible realizar ***herencia múltiple*** en base a clases. Sin embargo, es posible "simular" la herencia múltiple en base a las ***interfaces***." [13]

2.5.3. Interface

Una *interface* es un conjunto de declaraciones de métodos. Lo que una *interface* hace es sentar un parámetro que las clases que la usan deban seguir. Es decir, si una interface posee dos métodos distintos, la clase que implemente esta *interface* debe también contener estos dos métodos. Una *clase* puede implementar más de una *interface*, representando una forma alternativa de la herencia múltiple.

2.5.4. Package

Un ***package*** es una agrupación de clases. Los *packages* son utilizados por los usuarios para agrupar clases que estén relacionadas entre sí y así poder usarlas en todo el programa o en otros programas. Además de los *packages* creados por el usuario, existen *packages* propios de Java que son de mucha utilidad al momento de realizar una aplicación o un programa.

CAPÍTULO 3

PROGRAMACIÓN EN JAVA DEL PROGRAMA PLACA

En este capítulo se tratan temas relacionados con la programación de Placa, se presenta una breve introducción al programa, sus características y objetivos. Se trata sobre la programación en sí del programa, se explica los distintos ficheros que componen el programa, y también se explica las nuevas clases incorporadas en la versión 1.1 de Placa, así como sobre las clases de la versión 1.0 que se mantuvieron. Finalmente, se brinda una explicación sobre las librerías externas utilizadas, se realiza un ejemplo para mostrar la utilización del programa, y sobre la forma de crear el instalador de Placa.

3.1. Introducción

El programa Placa es un programa para uso didáctico y de investigación. En sus primeras dos versiones el programa resuelve problemas de esfuerzos estáticos lineales y problemas de transferencia de calor, específicamente problemas de conducción bidimensional con efectos de convección y generación interna.

Para que este programa cumpla con la finalidad didáctica y de investigación debe cumplir ciertos requisitos que lo hagan un programa fácil de entender y manejar, así como fácil de interpretar y observar los resultados. Por lo tanto Placa debe ser un programa con las siguientes características:

- Debe tener un interfaz gráfica para el uso intuitivo del usuario
- Capacidad de resolver problemas que se puedan discretizar con elementos tipo placa y elementos tipo 2D (placas planas) (Versión 1.0), y problemas discretizables con elementos triangulares y rectangulares (Versión 1.1)
- Discretización del elemento de acuerdo a un mallado estructurado y no estructurado en dos dimensiones
- Presentación de resultados en forma gráfica y numérica

3.2. Programación en Java

La versión 1.0 del programa Placa fue desarrollada utilizando en primer lugar MATLAB para posteriormente emplear J2SE Development Kit 5.0, Java 3D 1.3.1 y el entorno de programación NetBeans 5.0. Mientras que la versión 1.1 fue desarrollada empleando J2SE Development Kit 5.0 Update 9, Java 3D 1.5.0_01 y el entorno de programación NetBeans 5.0.

Por ser una versión nueva, algunas características de la versión 1.0 se mantuvieron sin ninguna modificación para no afectar el análisis de esfuerzo estático lineal, pero por otro lado se tuvieron que añadir nuevas clases propias del análisis de transferencia de

calor. De esta manera se garantiza que no existan conflictos entre los dos tipos de análisis y que cada uno trabaje en base a sus propias clases y compartiendo otras clases modificadas de la primera versión de Placa. Por esta razón los diagramas de flujo que describen los métodos más importantes que son, *mallaEstructurada, mallaNoEstructurada e iniciarAnalisis,* métodos que permiten realizar el mallado de la geometría y el posterior análisis, utilizados tanto en Placa V 1.0 como en Placa V. 1.1, no se incluyen en el presente trabajo; si se requiere mayor información se puede consultar la referencia [12].

En las tablas 3.1 y 3.2 se encuentra la información de las clases que fueron modificadas y añadidas en la versión 1.1 del programa Placa.

Tabla 3.1. Clases modificadas en Placa Versión 1.1.

Clases Modificadas	**Razón**
Placa	Por ser la clase principal, se debían añadir nuevos menús, nuevos botones, variables, nuevos métodos propios del análisis de transferencia de calor. De igual manera se modificó la parte visual de la ventana.
Acerca	Por ser la ventana que muestra la información general del programa, se debía añadir información sobre las nuevas modificaciones del programa así como los datos del autor.
CambiosAyuda	Por ser la clase que muestra los cambios realizados al programa, se debía añadir la información de las modificaciones realizadas como tipos de análisis que puede resolver, tipo de elementos, características añadidas.
ContenidoAyuda	Por ser la clase de ayuda del programa, debía contener la nueva información de ayuda para el tipo de análisis de transferencia de calor.
LibreriaMateriales	Por ser la clase que permite ver, crear o borrar materiales, se debía insertar nuevos campos que permitan ingresar el valor de la conductividad térmica, así como crear nuevos métodos que permitan guardar este nuevo dato.
Propiedades	Esta clase permite definir propiedades para las partes creadas. Por esta razón se debía añadir un campo para la conductividad térmica, propiedad indispensable para el análisis de transferencia de calor.
Unidades	Clase que permite definir el sistema de unidades del problema. Se modificó porque en la nueva versión se debía definir las unidades para las temperaturas, flujos de calor y conductividad térmica.
ArbolContenido	Clase requerida por *ContenidoAyuda*; se añadieron nuevas categorías que contenían información de ayuda para los pasos a seguir en el análisis de transferencia de calor.
ArbolModelo	Clase que muestra el árbol del modelo; se añadieron nuevas categorías como Temperaturas, Flujos, Generación y Convección.
MaterialesLista	Clase que contiene la lista de los materiales con sus respectivas propiedades. Se debió añadir a la lista de los materiales el valor de la conductividad térmica en unidades SI.
MaterialesUsuario	Clase que permite crear materiales propios; se necesitaban variables nuevas que definan la conductividad térmica.
Modelo	Clase que guarda todos los datos del problema a analizar. Se agregaron nuevas variables, se modificaron los métodos para guardar los datos y resultados dependiendo del tipo de análisis que se esté empleando.
RendererModelo	Clase que permite mostrar diferentes íconos en el árbol gráfico. Se añadieron variables para mostrar los íconos del árbol en el análisis de

	transferencia de calor.
Modelo3D	Clase que permite generar una geometría en 3D y permite manipularla; se crearon variables para poder manipular las temperaturas y flujos asociados a la geometría.
ArbolGrafico	Clase que crea un árbol en donde se muestra información del modelo a analizar. Se añadieron variables para mostrar las condiciones de los problemas de transferencia de calor.
PanelModelo	Clase en donde se dibuja la geometría mallada, permite seleccionar nodos para añadir las condiciones de borde. Se crearon variables y procesos para poder añadir las condiciones de borde propias de la transferencia de calor.
PanelResultados	Clase que muestra los resultados del análisis, se crearon variables que contenían la información de los resultados de transferencia de calor y que permitían mostrarlos en pantalla.

Tabla 3.2. Clases añadidas en Placa Versión 1.1.

Clases Añadidas	Razón
AnalisisTrans	Se necesitaba una clase nueva que muestre las etapas de resolución y el tiempo de análisis para el nuevo análisis implementado.
Conveccion	Clase que permite ingresar los valores de coeficiente de convección y de temperatura de convección para los nodos seleccionados.
Flujos	Clase que permite asignar los flujos de calor en X o Y a los nodos seleccionados.
Generacion	Clase que asigna valores de generación interna a uno o más nodos seleccionados.
Temperaturas	Clase que permite asignar valores de temperatura en la frontera para los nodos seleccionados.
AnalisisModelo2	Clase que realiza el análisis de transferencia de calor; era necesario crear otra clase para evitar problemas con las variables del otro tipo de análisis.
ElementoRectangular	Clase que contiene las matrices de rigidez y las matrices de convección para los elementos rectangulares. Se necesitaba crear otra clase porque las matrices para los dos tipos de análisis son distintas, al igual que los tipos de elementos.
ElementoTriangular	Clase que contiene las matrices de rigidez y las matrices de convección para los elementos triangulares. Se necesitaba crear otra clase porque las matrices para los dos tipos de análisis son distintas, al igual que los tipos de elementos.

El programa consta de 59 ficheros fuente, 22 ficheros de ayuda de extensión *htm* y 144 imágenes de extensión *png* distribuidos en 6 *packages*. Además, se emplea las librerías de distribución gratuita llamadas *colt* versión 1.2.0 y *mtj* versión 0.9.6. Ambas librerías permiten operar con matrices llenas y dispersas. Finalmente, el programa gratuito *IzPack* versión 3.10.2 crea el instalador para el programa.

A continuación se detalla el funcionamiento del programa, las distintas opciones que se manejan y mediante la ayuda de un ejemplo se muestran las distintas etapas para realizar un análisis usando Placa Versión 1.1. En el Anexo 1, se detallan las clases utilizadas en el programa, tanto las que requieren interfaz gráfica como las que no, se especifica las clases que fueron modificadas para la versión 1.1 del programa, así como las clases añadidas a esta misma versión de Placa. Las clases correspondientes a la

versión 1.0 del programa que no requieren interfaz gráfica y que no fueron modificadas pueden ser consultadas en la referencia [12].

3.2.1. Funcionamiento de Placa

Al iniciar Placa, la ventana principal muestra el área de dibujo de la geometría y el árbol dinámico en la parte izquierda de la misma. El análisis predeterminado con el que el programa arranca es el análisis de esfuerzos estático lineales. En esta instancia del programa, el usuario tiene la posibilidad de cambiar el tipo de análisis o de continuar con el análisis predeterminado. Posteriormente el usuario debe crear una nueva parte; al hacerlo se activan las distintas opciones de dibujo, como la creación de líneas, rectángulos, círculos y aparece una grilla en el área de dibujo. Si por descuido u olvido del usuario se dibuja la geometría pero el tipo de análisis no es el deseado, el usuario tiene la posibilidad de cambiar el tipo de análisis pero la parte creada se borrará. Una vez creada la parte, se puede seguir creando las partes que uno desee las mismas que se van añadiendo al árbol Geometría que se encuentra en la parte izquierda de la pantalla como a las ramas del mismo. El usuario tiene la posibilidad de borrar y modificar la parte creada haciendo clic en la rama del árbol con el nombre de la parte. Dentro de las características de la parte que se puede modificar están las propiedades físicas y térmicas, se puede también escoger un tipo de material de la librería de materiales y cambiar el espesor de la parte creada.

Una vez creada la parte y modificadas sus propiedades, el siguiente paso es realizar el mallado de la pieza. Para realizar el mallado el usuario tiene la opción de acceder a la ventana de Malla haciendo clic en el árbol en la sección de malla o por la barra de menús en la opción Modelo. Para el mallado el usuario tiene la opción de construir mallas estructuradas con elementos rectangulares y triangulares. Esta opción solo funciona para partes de forma rectangular. Para partes de forma arbitraria se pueden construir mallas no estructuradas con elementos triangulares. Además, el usuario determina el número aproximado de elementos que desea utilizar en la discretización a realizar así como la calidad de la malla, es decir, el usuario puede optimizar el proceso de mallado y mejorar la calidad de los elementos (triangulares) que conforman la malla, este mejoramiento de la malla se lo hace cambiando ciertas variables que más adelante se explican con más detalle.

Una vez mallada la geometría, y dependiendo del tipo de análisis a realizar, el usuario debe prescribir las condiciones de frontera del problema. Para el caso de esfuerzos estáticos el usuario debe definir apoyos y cargas, con la condición de que si no se define ningún apoyo el botón de analizar no se hará visible. Para el caso de transferencia de calor el usuario puede definir ya sea temperaturas, flujos de conducción, flujos de convección o generación de calor. Para este tipo de análisis no existe restricción al momento de no ingresar algún tipo de condición, pues una vez ingresada cualquier condición de frontera el botón de analizar se hará visible.

Con las condiciones de frontera definidas, el último paso consiste en realizar el análisis respectivo. Para esto se puede hacer clic en el botón de analizar o simplemente presionar F5. Una vez completado el análisis se mostrarán los distintos resultados obtenidos, a través de los cuales se pueden navegar haciendo clic con el botón derecho en la ventana de resultados y escogiendo el resultado que se desea visualizar o accediendo por la barra de menús a la opción Resultados; el modelo se puede grabar para tener acceso posterior al mismo. Finalmente, una vez guardado el modelo, ya sea

este de Esfuerzo Estático Lineal o de Transferencia de Calor, el usuario tiene la posibilidad de abrir el modelo sin necesidad de tener que cambiar el tipo de análisis primero. Es decir, si el usuario desea abrir un modelo de transferencia de calor no tiene que cambiar el tipo de análisis en el menú, sino que automáticamente cuando abre el modelo, la pantalla y los íconos correspondientes al tipo de análisis del modelo guardado se cargan en la ventana principal de Placa. De la misma forma puede abrir un modelo de Esfuerzo estático lineal estando abierto un modelo de transferencia de calor o simplemente si se encuentra en el tipo de análisis correspondiente a transferencia de calor. El usuario tiene la facultad de cambiar condiciones de un modelo guardado y realizar un nuevo análisis de este modelo, y guardarlo con las nuevas condiciones.

3.2.2. Ejemplo de funcionamiento de Placa Versión 1.1

El siguiente ejemplo muestra paso a paso el funcionamiento del programa Placa Versión 1.1, las distintas opciones, el aspecto de las ventanas cuando se escoge una opción determinada, los pasos que se deben seguir para realizar un análisis de transferencia de calor en estado estable, y finalmente la opción de guardar un modelo ya analizado. El ejemplo que se va a utilizar para mostrar el funcionamiento de Placa fue tomado del libro *Introduction to Heat Transfer* de Incropera y DeWitt [4].

Un horno industrial es soportado por una larga columna hecha de ladrillo refractario, con una conductividad térmica (k = 1 W/mK), con dimensiones de 1 m x 1 m. Durante condiciones de estado estable, tres superficies de la columna se mantienen a 500K mientras que la superficie restante se expone a un flujo de aire cuya temperatura es de 300K y su coeficiente de convección (h = 10 W/m^2K). Se pide encontrar la distribución de temperaturas en la columna.

La siguiente figura muestra la geometría del problema y las condiciones de frontera:

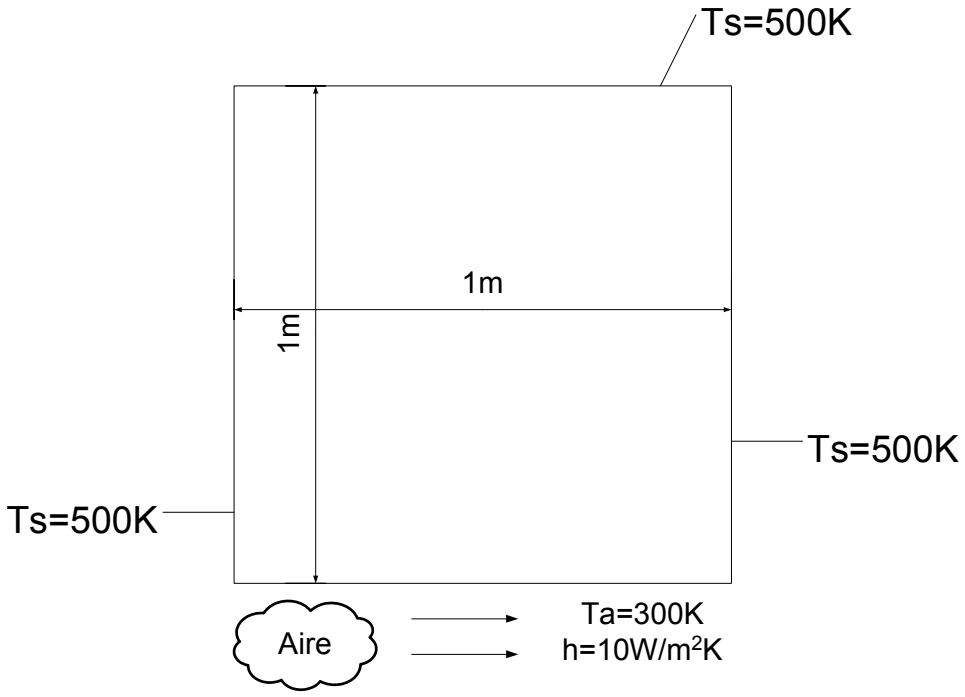


Figura 3.1. Geometría y condiciones de frontera ejemplo de visualización.

Para empezar el análisis se corre Placa mostrando la ventana inicial como se observa en la siguiente figura:

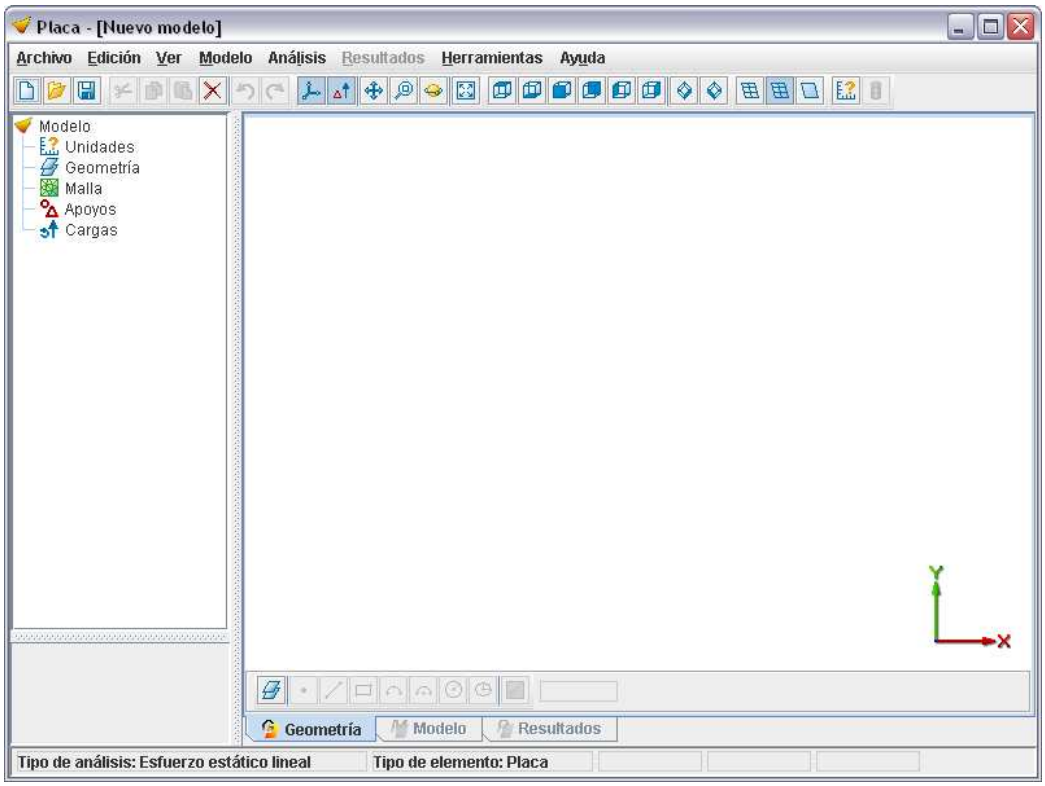


Figura 3.2. Ventana inicial de Placa.

Como el tipo de análisis predeterminado es el de esfuerzo estático lineal, se debe cambiar primero el tipo de análisis. Entonces en la barra de menús en la opción Análisis, se encuentra una opción que dice Tipo de Análisis, dentro de este submenú están las opciones Esfuerzo estático lineal y Transferencia de Calor estado estable, al hacer clic en esta opción automáticamente se cambia el tipo de análisis y la ventana inicial de Placa cambia su aspecto. En la figura 3.3 se muestra como cambiar el análisis y en la figura 3.4 la ventana de Placa una vez hecho el cambio.

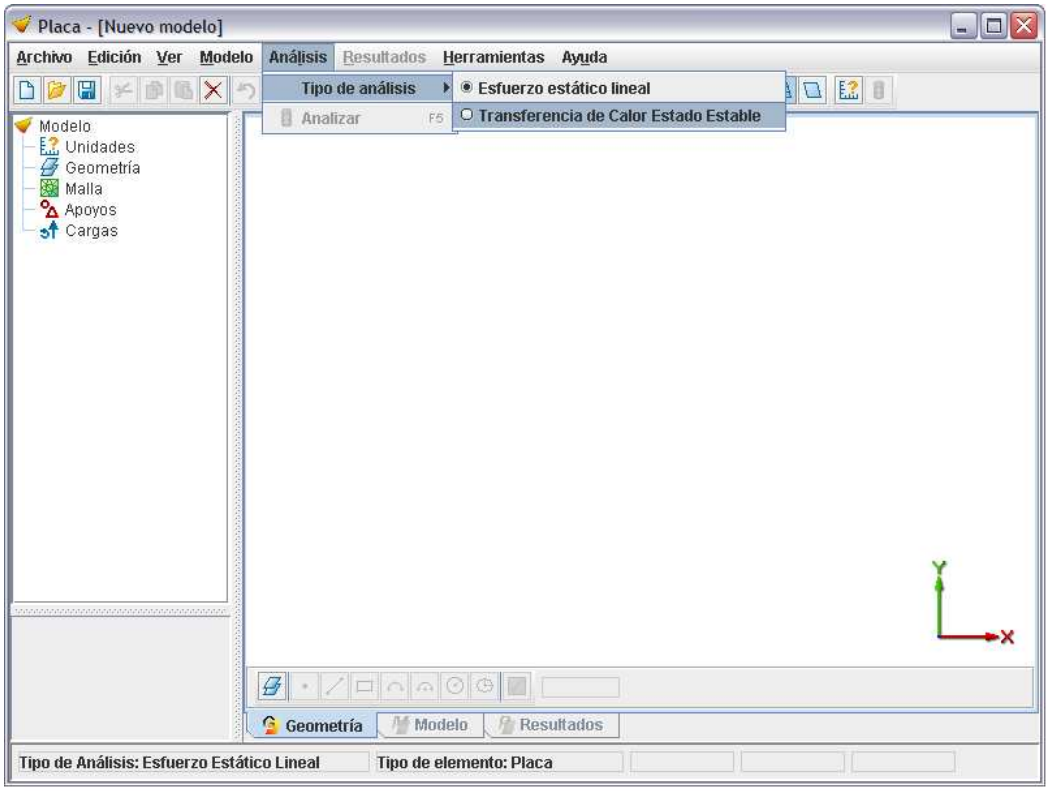


Figura 3.3. Cambio del tipo de análisis.

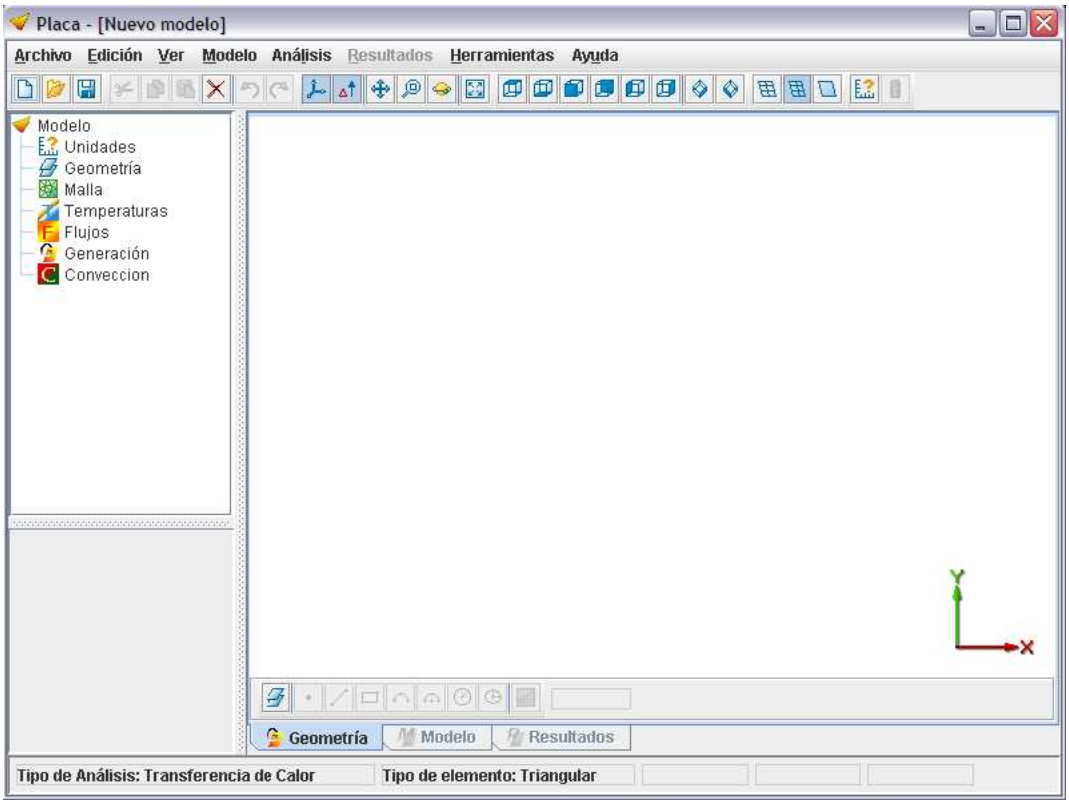


Figura 3.4. Ventana de Placa para el análisis de transferencia de calor.

Posteriormente, una vez hecho el cambio de tipo de análisis se crea una nueva parte y se dibuja la geometría del problema usando las herramientas de dibujo. Para crear la geometría del problema se sigue los siguientes pasos:

Se selecciona el botón de nueva parte:

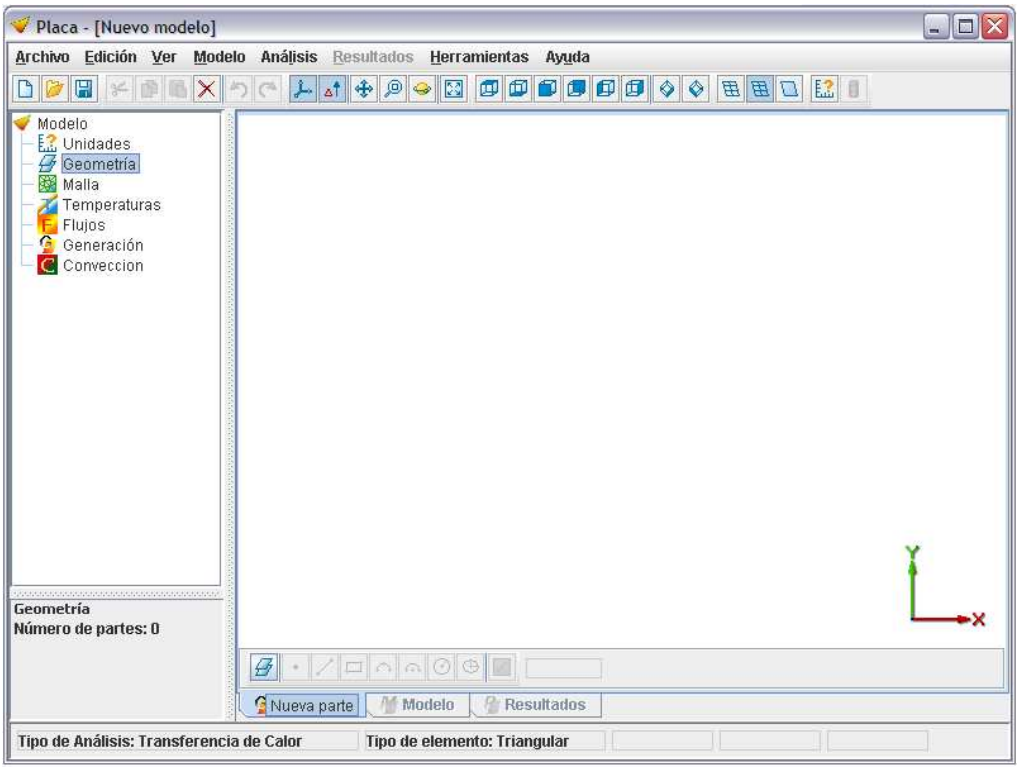


Figura 3.5. Botón para crear una nueva parte.

Al hacer clic en el botón de nueva parte aparece una grilla y las herramientas para dibujar.

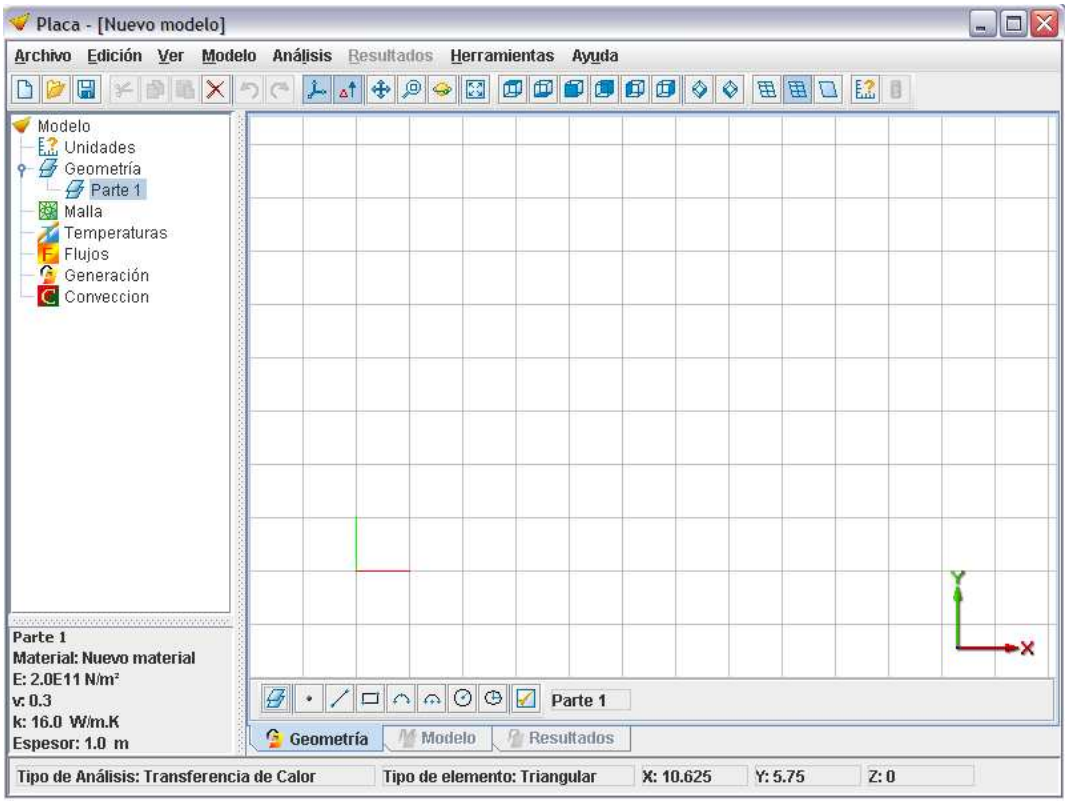


Figura 3.6. Ventana para dibujar geometrías.

Se dibuja la geometría en la zona de la grilla, usando los comandos de rectángulo, círculo, arco o línea, para este caso se usó el rectángulo; se especifica las coordenadas o se dibuja directamente sobre la grilla arrastrando el ratón y se obtiene el cuadrado deseado.

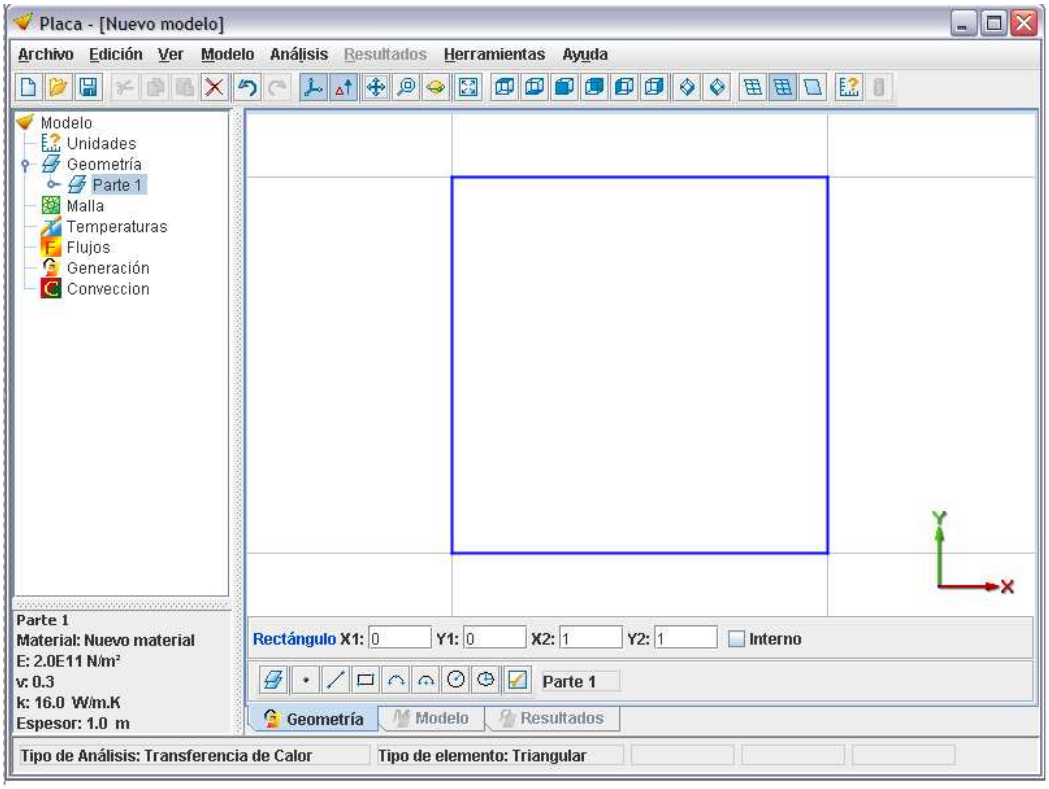


Figura 3.7. Geometría dibujada en Placa.

Cuando la geometría está creada, el siguiente paso es discretizar este dominio. El discretizado o mallado se lo puede hacer de dos maneras, la una haciendo clic en el árbol de lado izquierdo de la ventana o por la barra de menús en la opción Modelo.

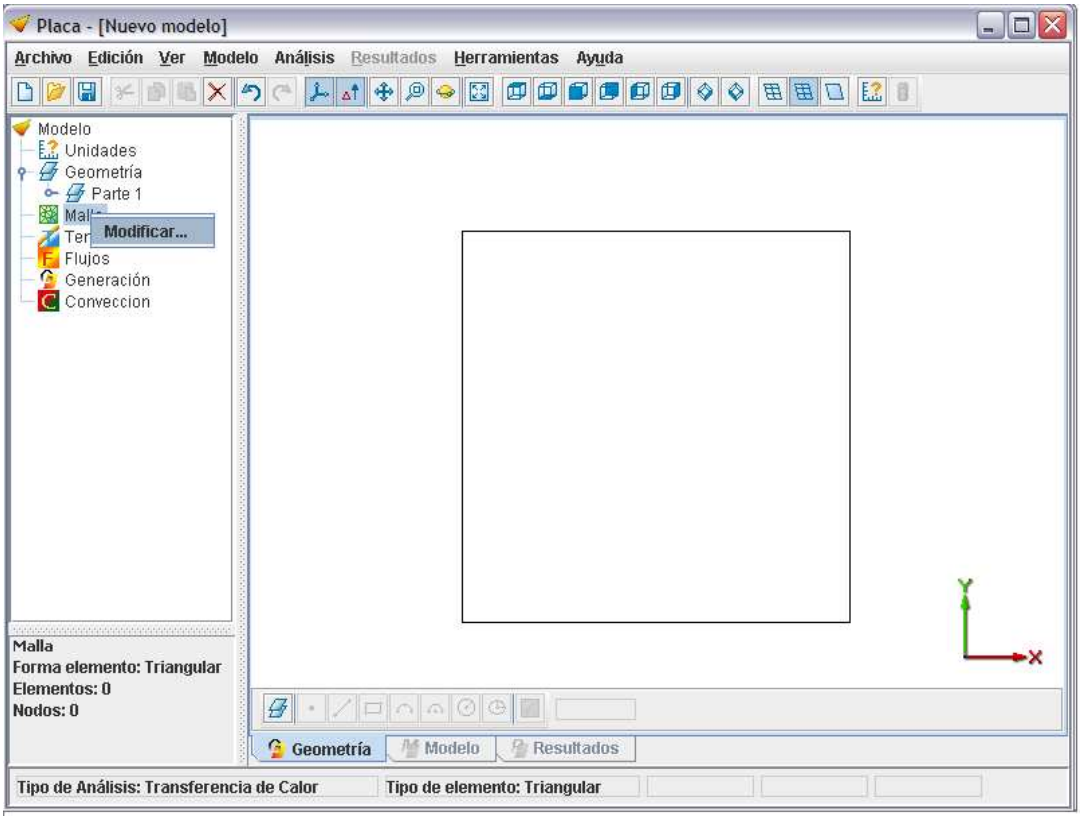


Figura 3.8. Acceso a malla por el árbol.

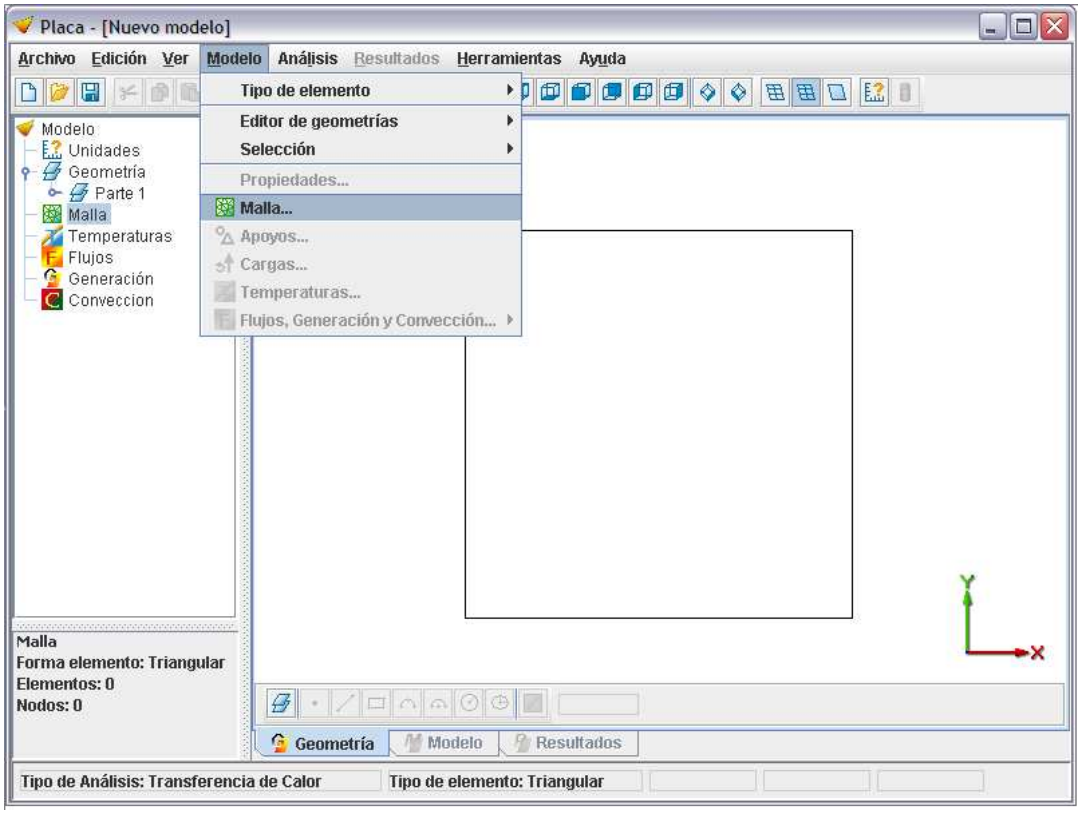


Figura 3.9. Acceso a malla por la barra de menús.

Al hacer clic en malla aparece otra ventana que permite crear la malla, esta puede ser estructurada o no estructurada, se especifica el número de elementos de la misma, así como otros factores que mejoran la calidad del mallado. En el Anexo A se explica con más detalle cada uno de los factores de la siguiente ventana.

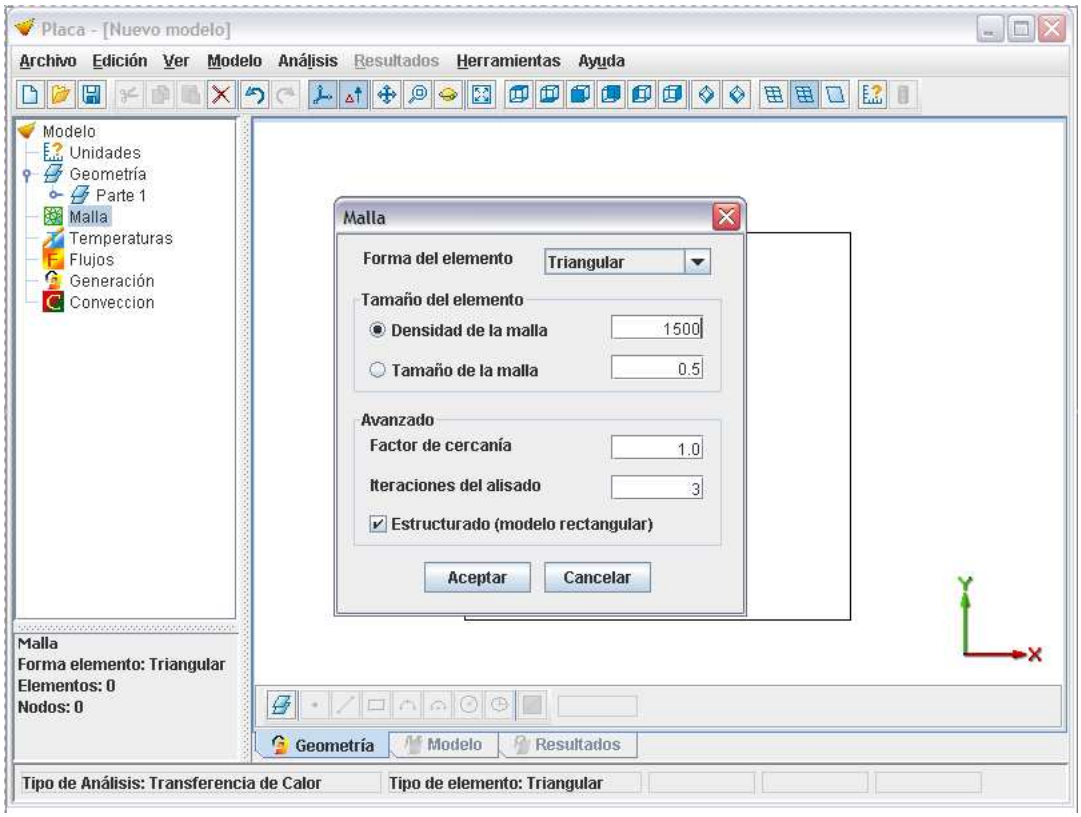


Figura 3.10. Ventana Malla.

En la figura 3.11 se muestra la geometría mallada con 1500 elementos aproximadamente y con una malla de elementos triangulares dispuestos estructuradamente. Por otra parte, en la figura 3.12 se muestra la misma geometría pero con una malla de elementos triangulares no estructurada.

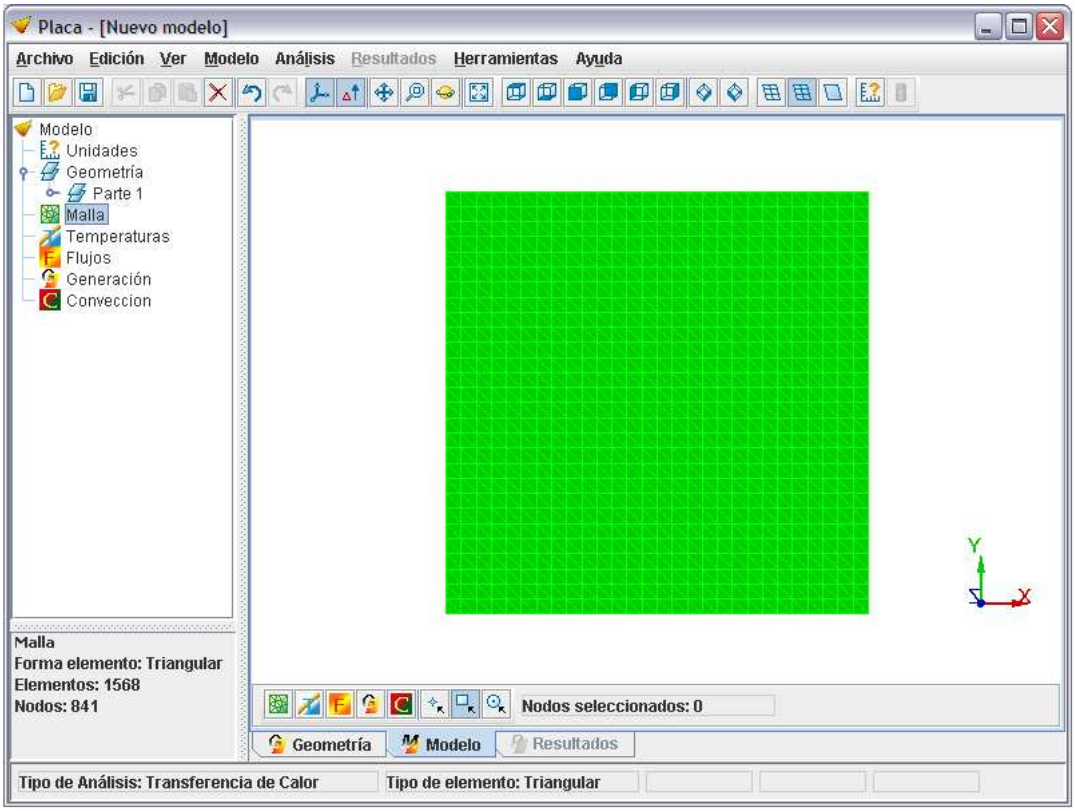


Figura 3.11. Geometría con malla estructurada.

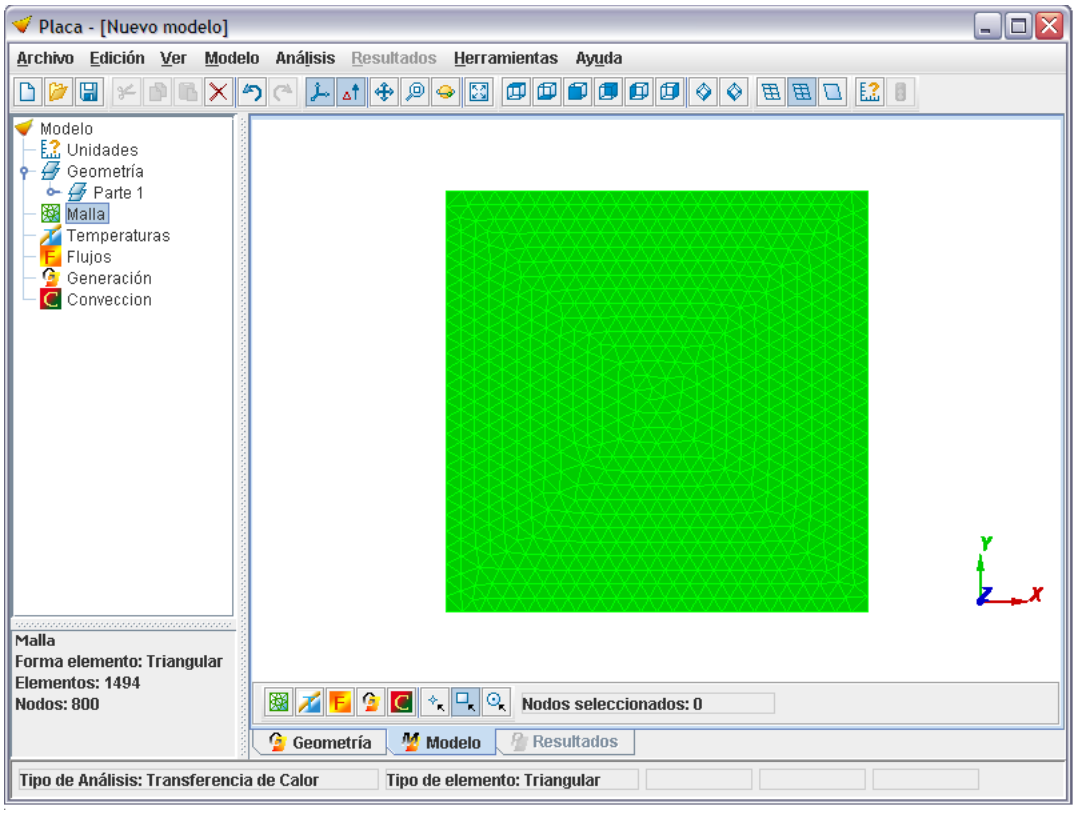


Figura 3.12. Geometría con malla no estructurada.

Ya discretizado el dominio, lo siguiente es ingresar las condiciones de frontera. Para este ejemplo se debe especificar las temperaturas de las superficies y el flujo de convección presente. La figura 3.13 muestra la geometría con sus condiciones ya especificadas. Las flechas amarillas indican la existencia de un flujo convectivo mientras que los puntos rojos indican una temperatura presente.

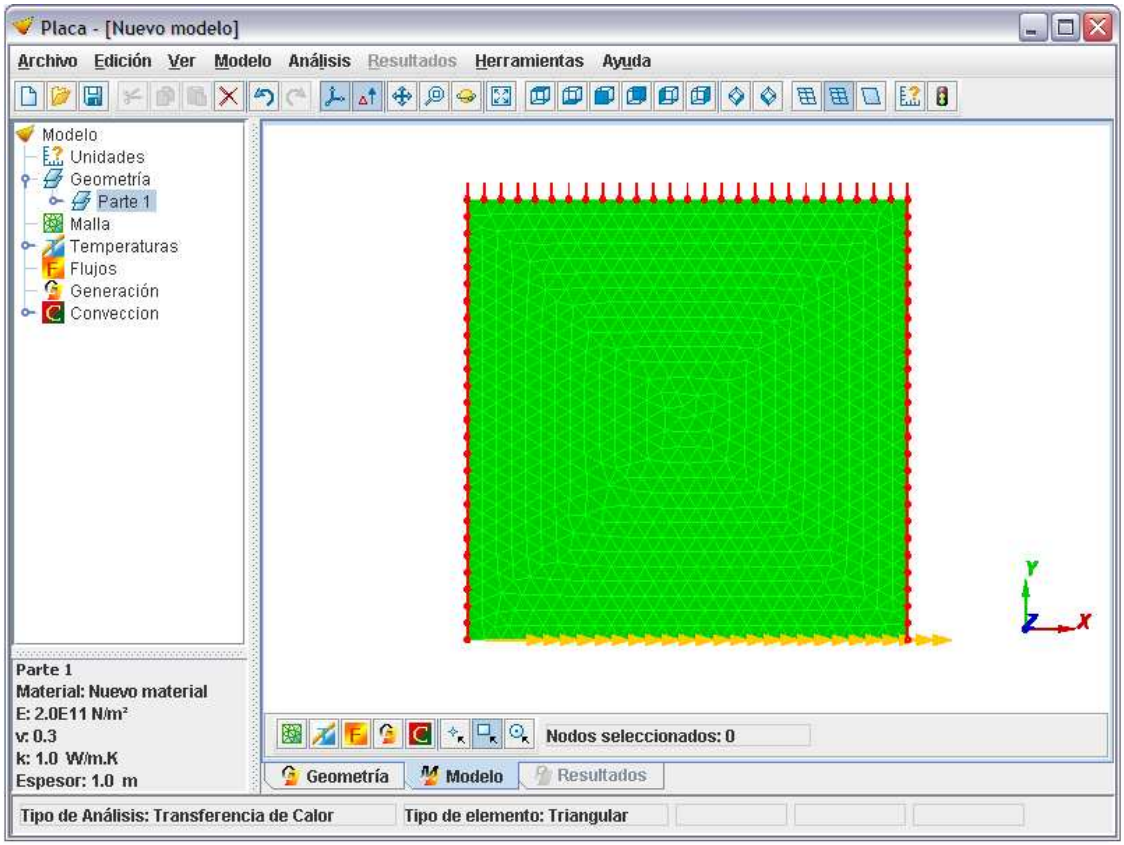


Figura 3.13. Geometría con condiciones de frontera especificadas.

Cuando ya se tiene las condiciones de frontera establecidas, lo siguiente es analizar el problema. Haciendo clic en el botón de analizar, o presionando F5, el programa empieza a analizar el problema, mostrando una ventana que detalla el tiempo empleado y los pasos ejecutados por el resolutor hasta terminar el análisis; la figura 3.14 muestra esta ventana.

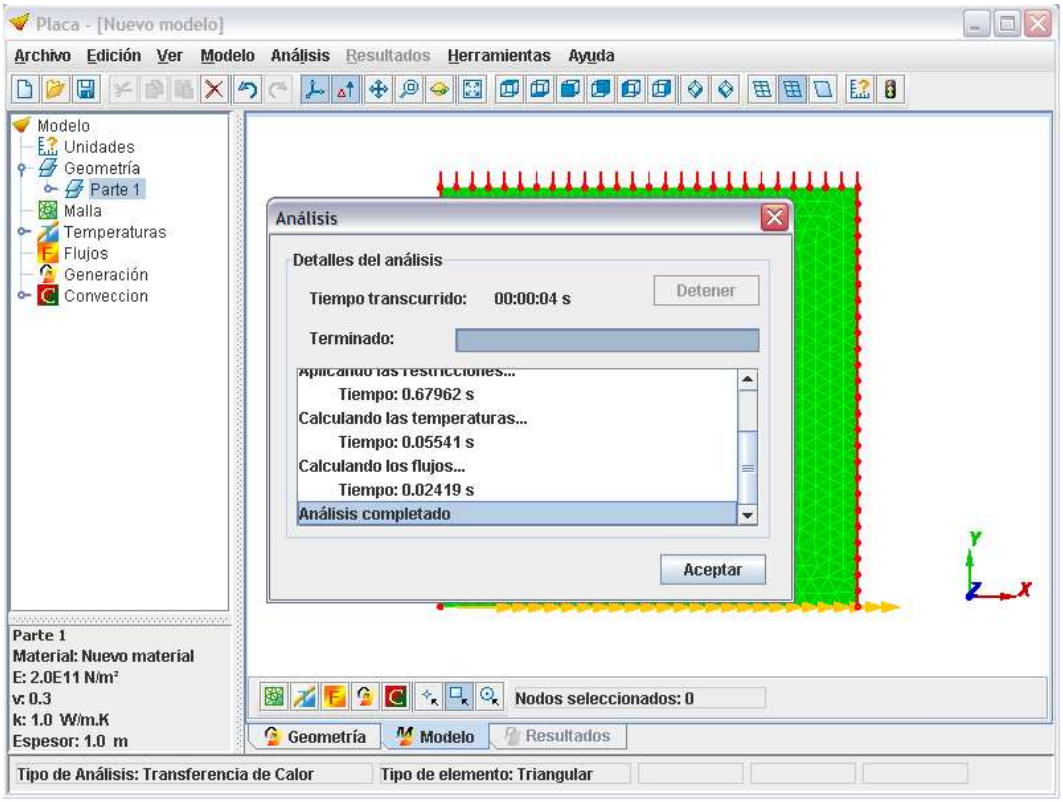


Figura 3.14. Ventana Análisis.

Terminado el análisis, Placa presenta los resultados obtenidos. En primera instancia muestra la distribución de temperaturas, pero al hacer clic derecho se puede navegar y seleccionar los demás resultados calculados para tener una apreciación gráfica de los mismos.

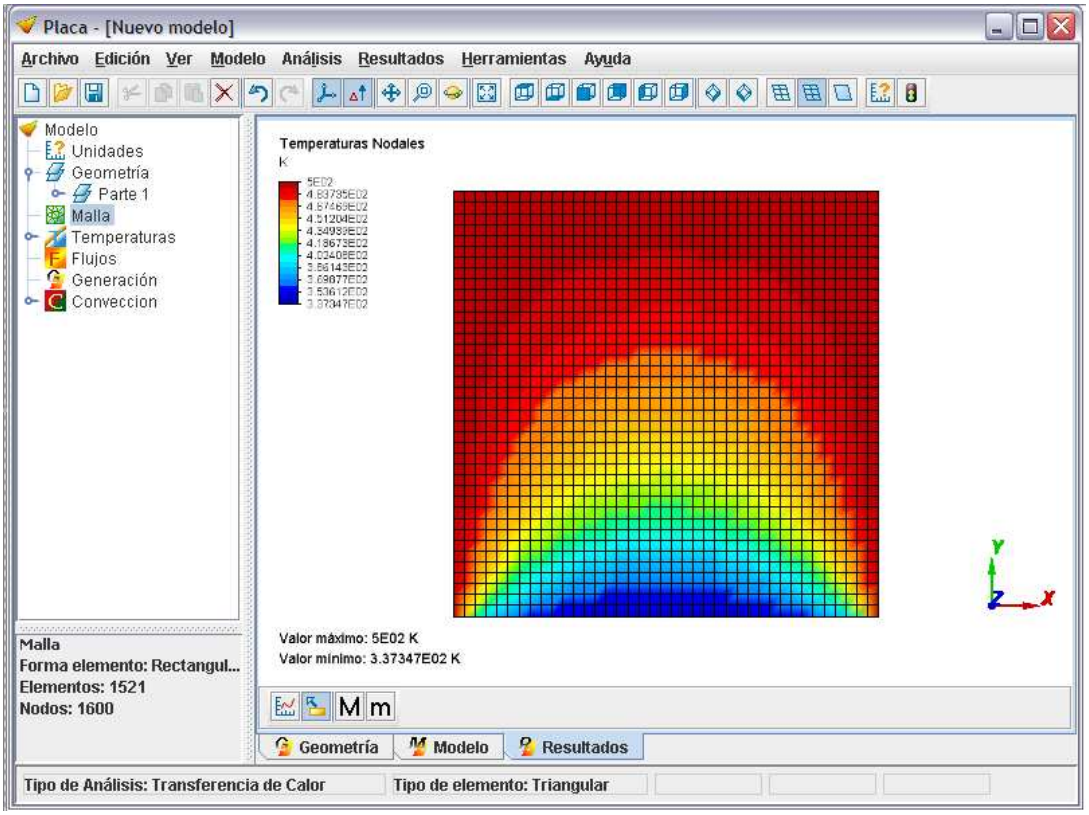


Figura 3.15. Distribución de temperaturas ejemplo de visualización.

Finalmente, ya mostrados los resultados se puede guardar el modelo para próximos análisis. También se puede guardar un archivo de texto que contenga los resultados obtenidos exclusivamente.

3.2.3. Ejecución del programa Placa

Para la ejecución del programa se requiere que en el equipo a utilizar esté instalada la máquina virtual de Java versión 5.0 o posterior y el módulo java3D versión 1.3.1 o posterior. Para la instalación es suficiente con dar doble clic al fichero *InstalarPlacaV1.1.jar* (solo válido en Windows) o escribir en una terminal de línea de comandos:

PATH_JAVA/java -jar PATH_INSTALADOR/InstalarPlacaV1.1.jar

Donde PATH_JAVA es la ruta hasta el ejecutable *java* y PATH_INSTALADOR es la ruta que indica donde está *InstalarPlacaV1.1.jar*. Una vez instalado, para iniciar el programa se puede dar doble clic al acceso directo del programa, o ejecutar *EjecutarPlaca.bat* en Windows y *EjecutarPlaca.sh* en Linux, para otras plataformas se debe escribir en una terminal de línea de comandos:

PATH_JAVA/java -Xmx256m -jar PATH_INSTALADOR/Placa.jar

3.2.4. Librerías externas usadas por el programa

El programa Placa, además de las librerías que se encuentran en el programa desarrollado, emplea 3 librerías de distribución gratuita, completamente desarrolladas en Java. A continuación se da una breve descripción de cada una.

colt: Es una librería que provee una infraestructura para la computación técnica y científica en Java. Contiene algoritmos para el análisis de datos, álgebra lineal, estadística, programación paralela y concurrente, entre otros. Estas librerías incluidas en colt se utilizan en la parte del análisis de los problemas, puesto que permiten manejar operaciones matriciales usando de una manera más eficiente la memoria utilizada haciendo de los cálculos más rápidos evitando la pérdida de tiempo y el posible colgamiento del sistema. Para más información se puede consultar su página web: http://dsd.lbl.gov/~hoschek/colt

mtj: Es una colección detallada de estructuras de datos de matrices, resolutores lineales, métodos de mínimos cuadrados y descomposiciones de vectores y valores propios. Está diseñada para ser usada como una librería para el desarrollo de aplicaciones numéricas. Permite que la aplicación corra más rápidamente al igual que los cálculos puesto que optimiza estos procesos de cálculos matriciales logrando un ahorro de tiempo y de memoria. Su página web es: http://rs.cipr.uib.no/mtj

IzPack: Es una herramienta que permite construir instaladores ligeros que se ejecuten en cualquier sistema operativo. Su diseño es modular, permitiendo seleccionar las aplicaciones que se instalarán para una plataforma específica. Su página web es: http://www.izforge.com/izpack

IzPack emplea archivos de extensión *xml* para describir la instalación. Para generar un instalador para el programa Placa V1.1 utilizando IzPack se debe seguir los siguientes pasos. Primero se debe crear los archivos *.xml* necesarios para la generación del instalador. Por ejemplo, el siguiente fragmento de código es del archivo *InstalarPlacaV1.1.xml*, el cual detalla las opciones para generar el instalador para el programa:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
<!--
   Instalador del programa Placa
   Creado con IzPack versión 3.10.2
-->
<installation version="1.0">
   <!--
      Sección de información
   -->
   <info>
      <appname>Placa</appname>
      <appversion>1.1</appversion>
      <authors>
         <author name="David Escudero Guevara"
               email="davidrescudero@hotmail.com"/>
      </authors>
            <javaversion>1.5.0 Update 9</javaversion>
   </info>
    <!--
      Preferencias de la instalación
            Instalador de dimensiones 640x480. con opción de cambio
   -->
```

```
<guiprefs width="640" height="480" resizable="yes">
        <laf name="looks">
        <os family="windows" />
        <param name="variant" value="plastic3D" />
        </laf>
        <modifier key="useButtonIcons" value="yes"/>
        <modifier key="useLabelIcons" value="yes"/>
        <modifier key="labelGap" value="2"/>
        <modifier key="layoutAnchor" value="CENTER"/>
        <modifier key="useHeadingPanel" value="yes"/>
        <modifier key="headingLineCount" value="1"/>
        <modifier key="headingFontSize" value="1.5"/>
        <modifier key="headingBackgroundColor" value="0x00ffffff"/>
        <modifier key="headingPanelCounter" value="text"/>
        <modifier key="headingPanelCounterPos" value="inHeading"/>
</guiprefs>
<!--
   Sección de localización
   Idiomas disponibles: español, inglés y francés
-->
<locale>
   <langpack iso3="spa"/>
        <langpack iso3="eng"/>
        <langpack iso3="fra"/>
</locale>
<!--
   Sección de recursos
-->
<resources>
        <res id="Installer.image" src="Placa.png"/>
        <res id="HTMLInfoPanel.info" src="LeameV1.1.htm"/>
        <res id="shortcutSpec.xml" src="AccesoWindows.xml"/>
        <res id="Unix_shortcutSpec.xml" src="AccesoUnix.xml"/>
</resources>
   <!--
   Sección de paneles
   Se usarán los paneles en el orden en que están indicados  a continuación
-->
<panels>
   <panel classname="HelloPanel"/>
   <panel classname="HTMLInfoPanel"/>
   <panel classname="TargetPanel"/>
   <panel classname="PacksPanel"/>
   <panel classname="InstallPanel"/>
        <panel classname="ShortcutPanel"/>
   <panel classname="SimpleFinishPanel"/>
</panels>
   <!--
   Sección de paquetes
-->
<packs>
   <pack name="Placa V 1.1" required="yes">
     <description>Programa compilado en NetBeans IDE 5.0 </description>
      <file src="TesisFinal.jar" targetdir="$INSTALL_PATH"/>
      <file src="LeameV1.1.htm" targetdir="$INSTALL_PATH"/>
                <file src="MaterialesUsuario.msr" targetdir="$INSTALL_PATH"/>
                <file src="bin" targetdir="$INSTALL_PATH"/>
                <file src="iconos" targetdir="$INSTALL_PATH"/>
                <file src="lib" targetdir="$INSTALL_PATH"/>
                <parsable targetfile="$INSTALL_PATH\bin\EjecutarPlaca.sh"/>
```

```
                <parsable targetfile="$INSTALL_PATH\bin\DesinstalarPlaca.sh"/>
                <executable targetfile="$INSTALL_PATH\bin\EjecutarPlaca.sh"stage="never"/>
                <executable targetfile="$INSTALL_PATH\bin\DesinstalarPlaca.sh"
stage="never"/>
            </pack>
        <pack name="Código Fuente" required="no" preselected="yes">
          <description>Código Fuente del programa Placa Versión 1.1</description>
          <file src="src" targetdir="$INSTALL_PATH"/>
        </pack>
    </packs>
    <!--
            Sección de archivos nativos
            Permite crear accesos directos en Windows
    -->
    <native type="izpack" name="ShellLink.dll"/>
    </installation>
```

Una vez creado este archivo, se debe crear una carpeta en la cual se guarden los archivos necesarios para crear el instalador, para el caso de Placa Versión 1.1 son necesarios los siguientes archivos:

- *iconos*.- carpeta que incluye los iconos del programa Placa y del desinstalador del programa.
- *src*.- carpeta que contiene los archivos fuentes (opcional).
- *lib*.- carpeta que contiene los archivos correspondientes a las librerías externadas usadas en Placa citadas anteriormente.
- *bin*.- carpeta que contiene archivos de ejecución del programa desde la consola de Windows.
- *InstalarPlacaV1.1.xml*.- archivo principal el cual genera el instalador de Placa V1.1.
- *AccesoWindows.xml y AccesoUnix.xml(opcional)*.- archivos que permiten la creación de accesos directos tanto para la plataforma Windows como para la plataforma Unix.
- *LeameV1.1.html*.- archivo que contiene la información básica del programa.
- *Materiales Usuario.msr*.- archivo que contiene la lista de materiales creados por el usuario.
- *TesisFinal.jar*.- archivo creado al momento del desarrollo del programa en NetBeans.

Una vez creada la carpeta con los archivos necesarios el siguiente paso es crear el instalador de Placa. Para esto se requiere usar la consola de Windows. Dentro de la consola se debe especificar primero la ruta en donde se encuentra la carpeta con todos los archivos, por ejemplo:

D:\Respaldos\David USFQ\Placa Versión 1.1>

Posteriormente se especifica la ruta en donde esté instalado Izpack, y se coloca entre comillas la ruta al archivo seguido de la palabra *compile y fuera de las comillas se coloca InstalarPlacaV1.1.xml.* La línea de comando de Windows debe visualizarse así:

D:\Respaldos\David USFQ\Placa Versión 1.1>"C:\Archivos de programa\IzPack\bin\ compile" InstalarPlacaV1.1.xml

Esta línea de código crea el archivo *InstalarPlacaV1.1.jar*. Para mayor información se puede consultar la ayuda incluida con *IzPack.*

CAPÍTULO 4

VALIDACIÓN DE RESULTADOS

En este capítulo se presentan las distintas pruebas realizadas para verificar el correcto funcionamiento y validar los resultados obtenidos mediante el programa Placa. Las pruebas se realizaron seleccionando varios problemas típicos de transferencia de calor. Estos problemas fueron analizados mediante otros dos programas comerciales y mediante la solución analítica en alguno de ellos, para de esta manera obtener una comparación y sacar las conclusiones del caso.

4.1. Introducción

Una vez finalizada la programación del programa Placa en su versión 1.1, es imperativo realizar algunas pruebas para verificar si los algoritmos implementados son correctos. Para esto se decidió seleccionar y analizar con Placa un conjunto específico de problemas para luego probar si el programa producía resultados coherentes con los resultados obtenidos con programas comerciales de elementos finitos y con las soluciones analíticas de algunos de los problemas seleccionados. Los programas utilizados para realizar las comparaciones fueron Algor versión 19 y Cosmos Works 2007. Se escogió estos dos software por dos razones principales. La primera fue la disponibilidad de ambos programas, por lo que no era necesario adquirir el software ni la licencia correspondiente para el uso de los mismos. La segunda razón fue la razón comercial: Algor es uno de los programas más usados hoy en día en el mundo para la simulación y el análisis mediante el método de los elementos finitos. En su página web www.algor.com se encuentra una lista de los clientes que usan hoy en día Algor y que son más de 20000 en todo el mundo. Por otra parte, Cosmos Works forma parte de la familia de Solid Works , empresa reconocida mundialmente por sus paquetes computacionales y muy utilizada para diseño y simulación en todo el mundo.

Las pruebas de verificación se realizaron con ejemplos sencillos pero completos. Esto es, se escogieron problemas con un grado de complejidad relativamente no muy alto pero con las condiciones necesarias para realizar análisis completos de transferencia de calor. Para esto, los problemas contienen flujos de convección, así como generación interna, flujos de calor, y condiciones de temperaturas prescritas.

4.2. Ejemplo de verificación 1: Pared de un horno sometida a convección

En este ejemplo se tiene una sección de una pared de un horno compuesta por dos tipos de material. Esta pared es enfriada con agua que pasa a través de tuberías con diámetro de 0.75 pulgadas. La sección de la pared tiene dimensiones de 36 x 12.25 $pulg^2$ y un espesor de 12 pulgadas. La conductividad térmica del primer material es de 1.1E-5 (Btu/h.pulg.°F) mientras que la del segundo material es 6.254E-4 (Btu/h.pulg.°F). Por la tubería pasa agua a una temperatura de 70°F y con un coeficiente de convección de 0.0012 (Btu/h.$pulg^2$°F). Las superficies superior e inferior de la pared tienen temperaturas prescritas de 1200°F y 125°F, respectivamente. El problema se muestra en la figura 4.1. Por ser un problema simétrico se va a considerar la tercera parte de la pared para realizar el análisis. El objetivo es encontrar la distribución de la temperatura y el flujo de calor en la pared.

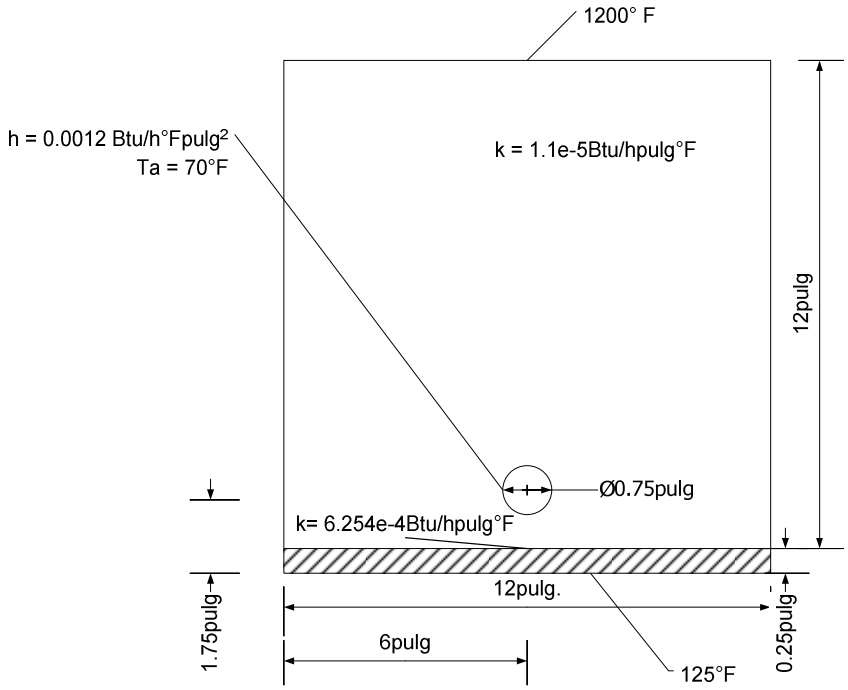


Figura 4.1. Dimensiones y condiciones del ejemplo 1.

4.2.1. Comparación de resultados

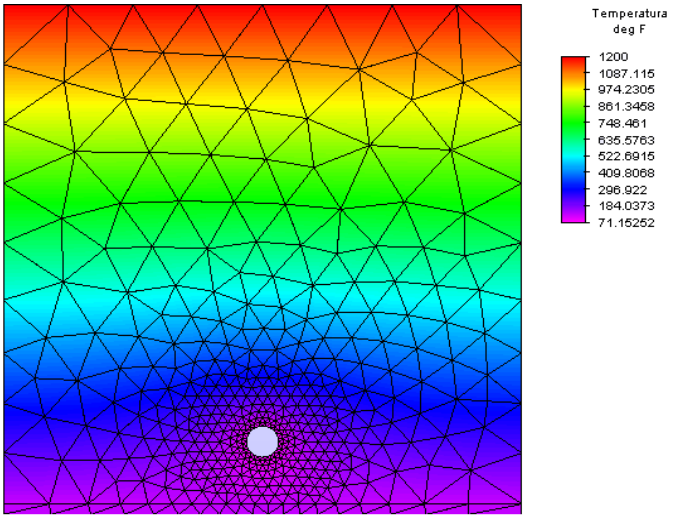
Una vez realizadas las simulaciones en los tres programas se van a comparar los resultados de la distribución de la temperatura y los flujos de calor por unidad de área. La tabla 4.1 muestra los resultados obtenidos para mallas de aproximadamente 800 elementos triangulares. Los resultados de los flujos son los valores máximos y mínimos obtenidos mientras que para este ejemplo se toma como referencia el valor de la temperatura obtenida alrededor de los tubos.

Tabla 4.1. Resultados obtenidos para el ejemplo 1.

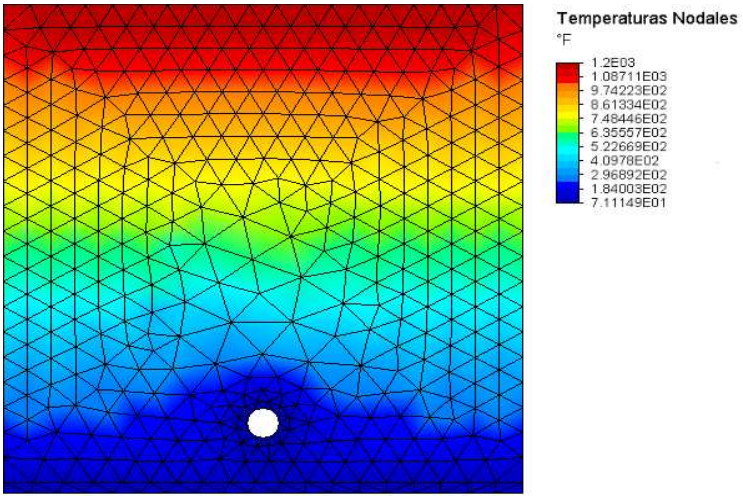
Programas	Algor	Placa	Cosmos Works
Forma de los elementos	Triangular	Triangular	Triangular
Número de elementos	829	810	818
Máx. flujo de calor (Btu/h·pulg^2)	0.0035217	0.002648	0.0036864
Mín. flujo de calor (Btu/h·pulg^2)	9.3735-5	4.5267e-6	7.8228e-6
Temperatura (°F)	71.1524	71.1149	71.19

En la figura 4.2 se presentan las distribuciones de temperatura obtenidas con cada uno de los programas usados. En primer lugar, se puede observar en la figura que existen algunas diferencias en el mallado que realizan los distintos programas. Tanto Algor como Placa muestran una malla compuesta por elementos triangulares pero dispuestos en forma no estructurada; es decir, elementos triangulares que no tienen simetría entre sí. En Cosmos Works siempre se van a presentar mallas con elementos triangulares pero dispuestos en una forma más estructurada, en la cual se observa mucha más simetría que en los anteriores dos programas.

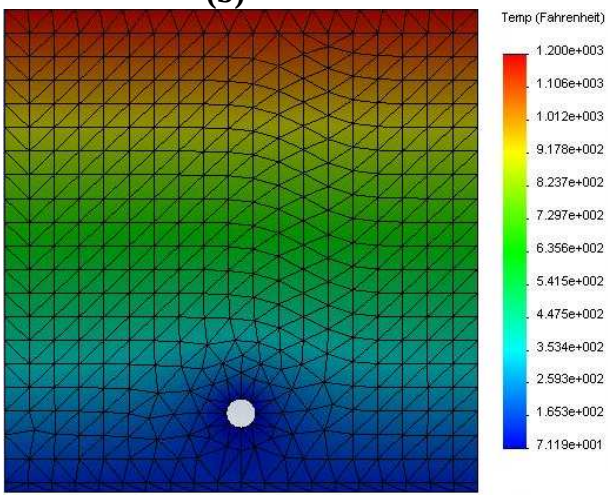
En segundo lugar, en la figura se puede apreciar que las distribuciones de temperaturas para los tres programas son prácticamente las mismas, con pequeñas diferencias dadas por el suavizado de las líneas de contorno que cada programa realiza al mostrar los resultados. En lo que respecta a los valores de las temperaturas mínimas obtenidas, estas son las mismas para los tres programas.



(a)



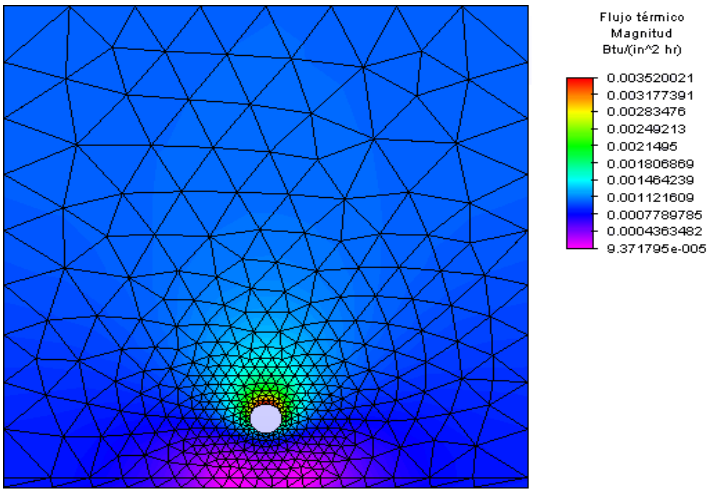
(b)



(c)

Figura 4.2. Distribución de temperaturas para el ejemplo 1: (a) Algor, (b) Placa, (c) Cosmos Works.

En la figura 4.3 se presentan las distribuciones de los flujos de calor.



(a)

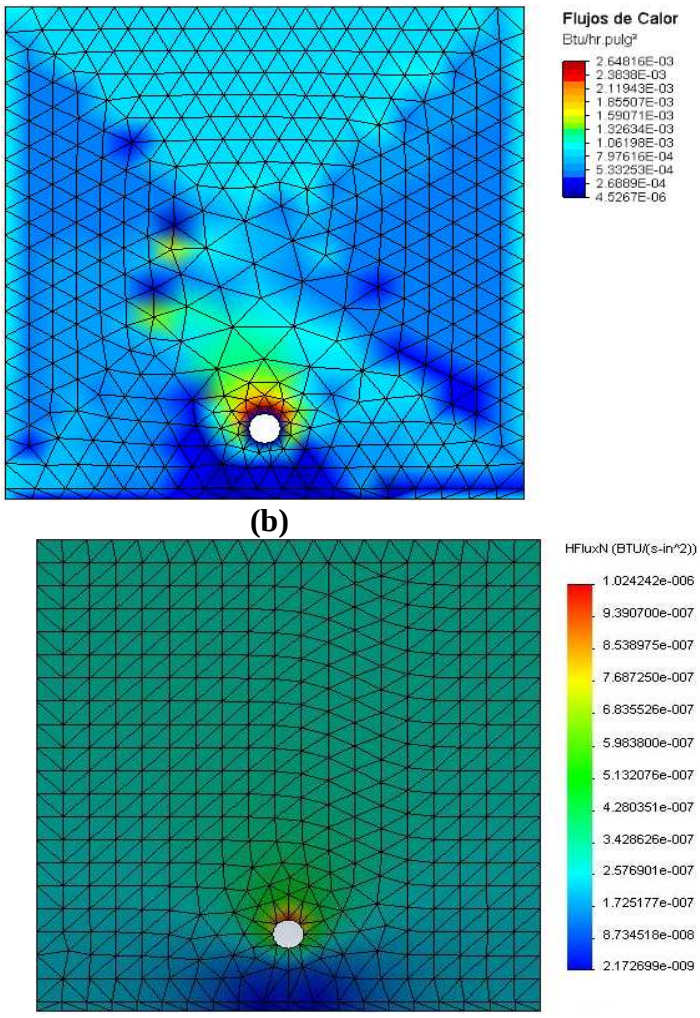


Figura 4.3. Distribución de flujos de calor para el ejemplo 1: (a) Algor, (b) Placa, (c) Cosmos Works.

En lo que respecta a los flujos de calor, se puede observar que tanto Algor como Cosmos presentan distribuciones parecidas. En el caso de Placa se puede observar que la distribución alrededor del agujero de la geometría es igual a la de los otros dos programas, pero se observa una variación de colores en el resto de la geometría que no concuerda con los resultados obtenidos en Algor y Cosmos. Como se puede apreciar en la figura 4.3 b) existen ciertos valores puntuales que evitan que la distribución hacia la parte superior de la placa sea uniforme observándose esas diferencias de colores entre las distribuciones. Esta variación en la distribución de los valores se debe a la forma de cálculo de los flujos de calor. Esto quiere decir que los flujos de calor de Placa se los calcula tomando cada nodo y comparando con los nodos adyacentes y comprobando si existe un flujo de calor, en cambio que para Algor y Cosmos los flujos se calculan partiendo del centroide del dominio.

4.3. Ejemplo de verificación 2: Chimenea industrial de concreto sometida a convección

Una pequeña chimenea industrial está hecha de un concreto que tiene una conductividad térmica de 1.4 W/mK. Las dimensiones de la chimenea son 0.6 x 0.6 x 5 m^3. La temperatura interna de la chimenea se supone uniforme y es igual a 100°C. La superficie exterior está expuesta al aire circundante, que se encuentra a una temperatura de 30°C y tiene un coeficiente de convección de 20 W/m^2K. Se busca determinar la

distribución de temperaturas en el concreto. En la figura 4.4 se puede apreciar las condiciones del problema y su geometría.

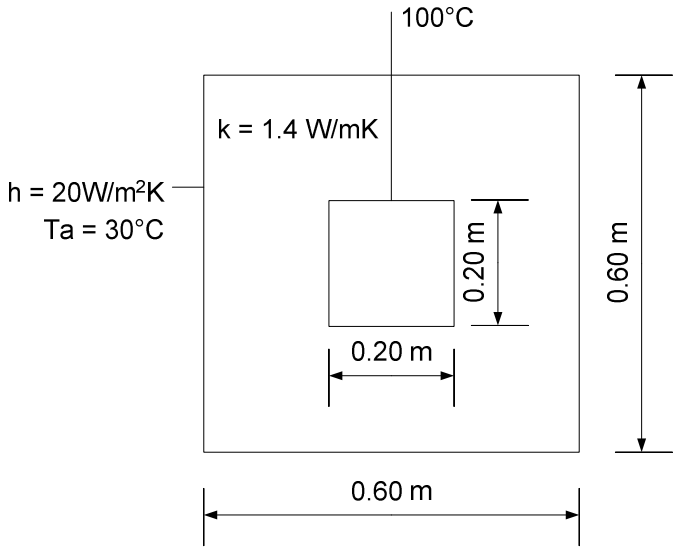


Figura 4.4. Geometría y condiciones del ejemplo 2.

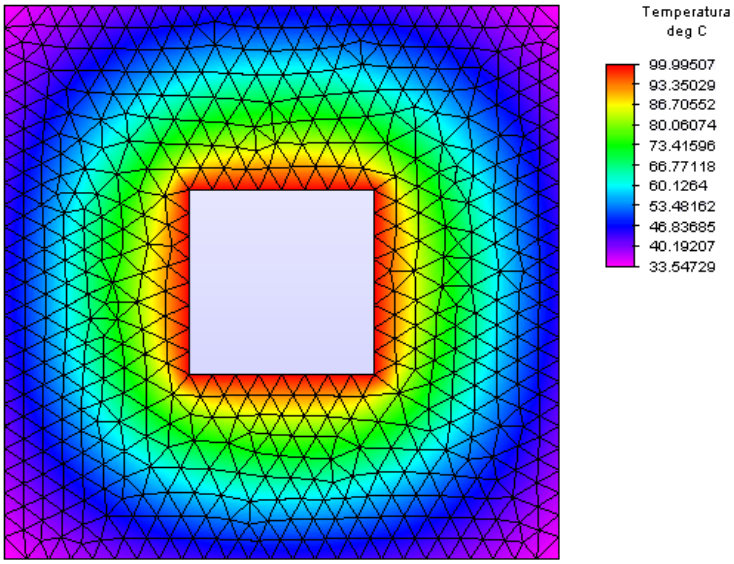
4.3.1. Comparación de resultados

La tabla 4.2 muestra los resultados obtenidos para mallas con varios elementos triangulares y muestra la solución analítica de este problema.

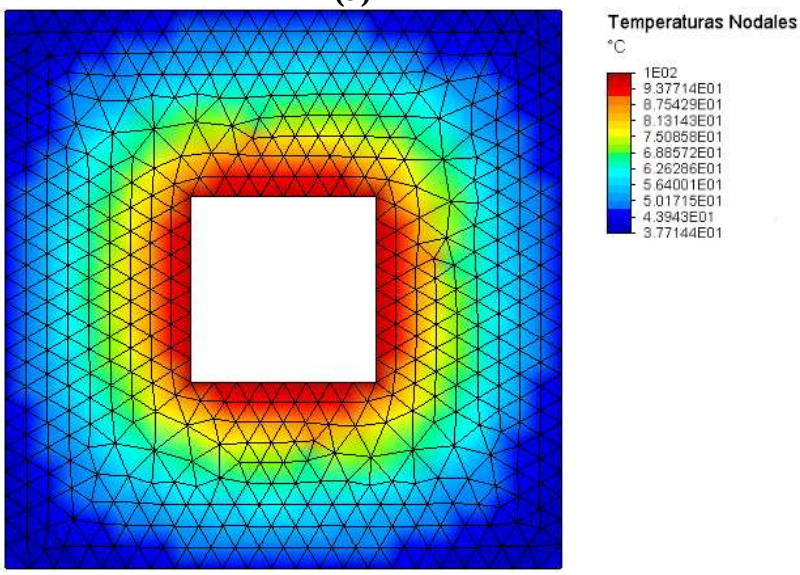
Tabla 4.2. Resultados obtenidos para el ejemplo 2.

Convergencia de Resultados Ejemplo 2						
Programa	Algor					**Solución Analítica[8]**
Forma de los elementos	Triangular					
Número de elementos	124	508	1058	2076	4988	
Temperatura máxima (°C)	100	100	100	100	100	100
Temperatura mitad chimenea (°C)	69	67.99	67.188	68.377	68.16	70.83
Temperatura mínima (°C)	33.0161	33.4651	33.5474	33.576	33.6	32.73
Programa	Placa					**Solución Analítica[8]**
Forma de los elementos	Triangular					
Número de elementos	112	502	1006	1974	4926	
Temperatura máxima (°C)	100	100	100	100	100	100
Temperatura mitad chimenea (°C)	70.365	69.02	68.857	68.269	68.356	70.83
Temperatura mínima (°C)	40.729	38.85	37.714	37.08	36.522	32.73
Programa	Cosmos Works					**Solución Analítica[8]**
Forma de los elementos	Triangular					
Número de elementos	120	488	1024	2042	5116	
Temperatura máxima (°C)	100	100	100	100	100	100
Temperatura mitad chimenea (°C)	67.525	66.81	65.62	67.60	67.01	70.83
Temperatura mínima (°C)	30	30	30	30	30	32.73

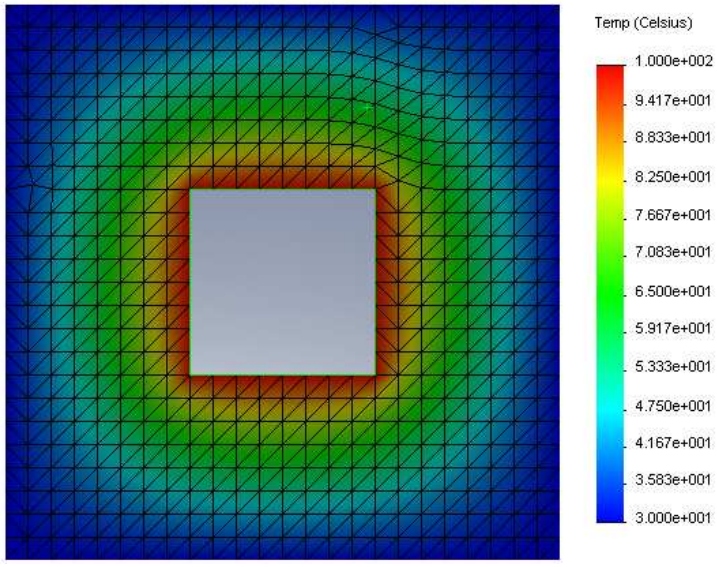
A continuación se presentan las distribuciones de temperatura obtenidas con cada uno de los programas usados para 1000 elementos



(a)



(b)



(c)

Figura 4.5. Distribución de temperaturas para el ejemplo 2: (a) Algor, (b) Placa, (c) Cosmos Works.

Como se puede apreciar en la figura 4.5, las distribuciones de temperatura obtenidas con Placa y Cosmos son similares, mientras que la distribución obtenida en Algor presenta una ligera diferencia en las esquinas exteriores de la geometría. Esta diferencia se debe a la distribución de colores que tiene Algor, ya que como se observa para Placa y Cosmos, los colores con tonalidades azules muestran los valores más bajos mientras que en Algor el color magenta representa los valores más bajos. Con respecto a los resultados numéricos y comparando con la solución analítica, se observa un porcentaje de error de aproximadamente el 10% en los valores mínimos tanto para Placa como para Cosmos Works, mientras que para Algor existe un porcentaje de error del 3% aproximadamente en el valor mínimo. Existe una diferencia de menos del 5% para los valores en la mitad de la chimenea comparados con la solución analítica. Estas variaciones en los valores se puede deber a la diferencia en el mallado de los diferentes programas. Al haber mallas diferentes, el número de nodos varía al igual que el número de elementos, lo que ocasiona una diferencia entre los resultados obtenidos.

4.4. Ejemplo de verificación 3: Motor de motocicleta

En este ejemplo se tiene la sección transversal de un motor de motocicleta hecho de una aleación de aluminio 2024-T6. La altura del motor es 0.15 m. Bajo condiciones normales de operación la superficie del motor se encuentra a una temperatura de 500K y está expuesta al aire a 300K con un coeficiente de convección de 50 W/m^2K. Las aletas del motor tienen una longitud de 20 mm y un ancho de 6 mm. Se busca encontrar la distribución de temperaturas y el flujo de calor del motor hacia el ambiente. La figura 4.6 muestra las condiciones del problema y su geometría.

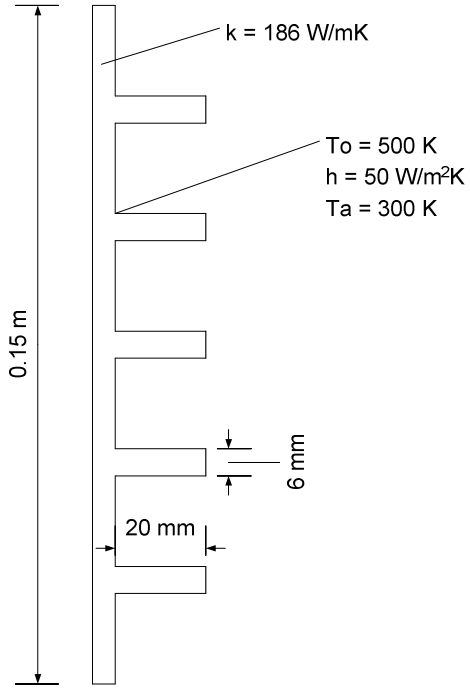


Figura 4.6. Geometría del ejemplo 3.

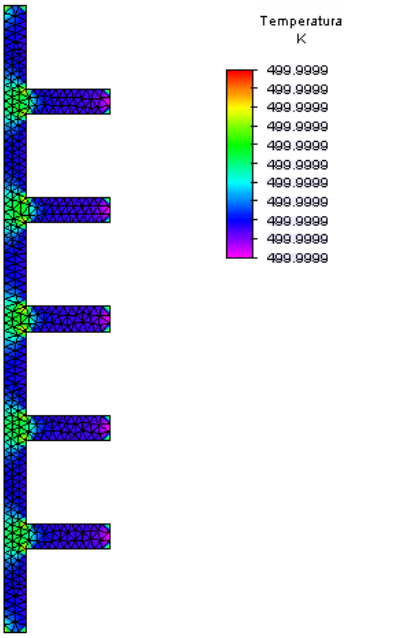
4.4.1. Comparación de resultados

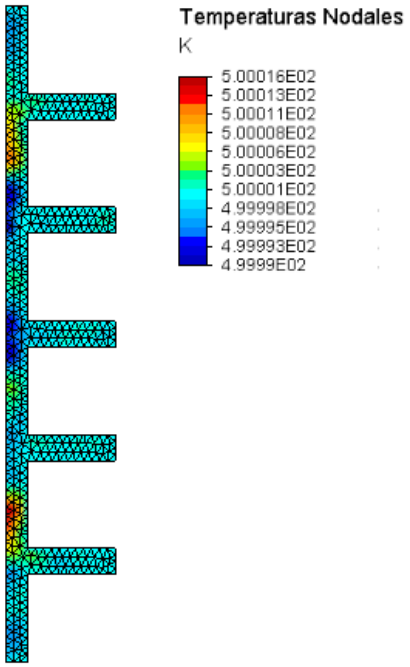
La tabla 4.3 muestra los resultados obtenidos para mallas de aproximadamente 750 elementos triangulares.

Tabla 4.3. Resultados obtenidos para el ejemplo 3.

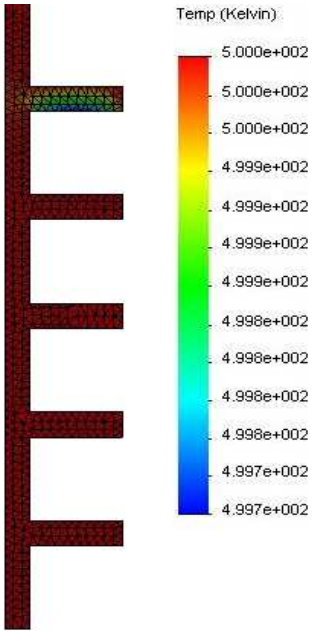
Programas	Algor	Placa	Cosmos Works
Forma elementos	Triangular	Triangular	Triangular
Número elementos	738	796	782
Temperatura máxima (K)	500	500	500
Temperatura mínima (K)	500	500	500
Flujo de Calor máximo (W/m^2)	8.20252	10.73	0.336

A continuación, en la figura 4.7, se presentan las distribuciones de temperatura obtenidas con cada uno de los programas usados.





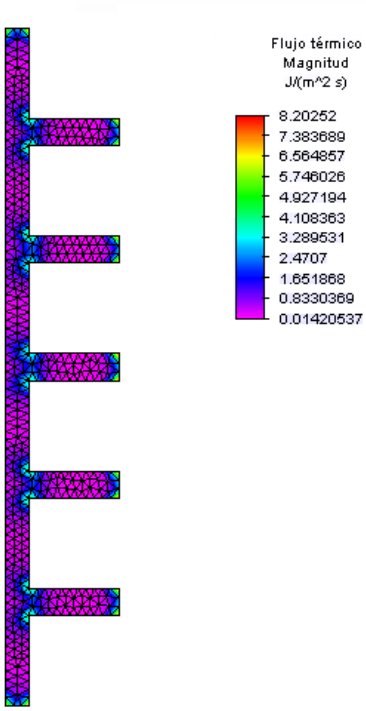
(a) **(b)**

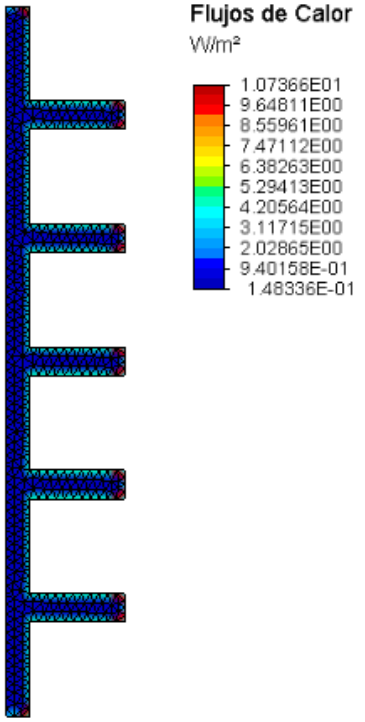


(c)

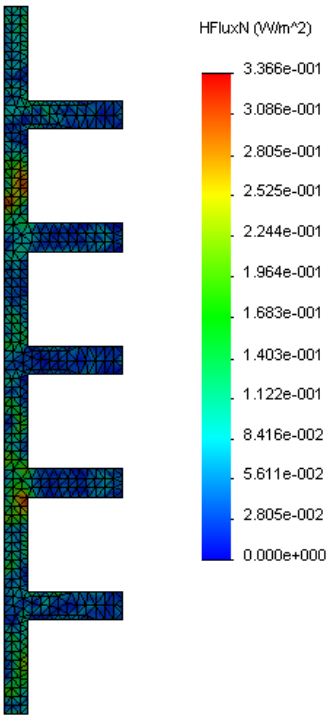
Figura 4.7. Distribución de temperaturas en el ejemplo 3: (a) Algor, (b) Placa, (c) Cosmos Works.

Por otro lado, las distribuciones del flujo de calor se presentan en la siguiente figura.


Flujo térmico
Magnitud
J/(m^2 s)
8.20252
7.383689
6.564857
5.746026
4.927194
4.108363
3.289531
2.4707
1.651868
0.8330369
0.01420537


Flujos de Calor
W/m²
1.07366E01
9.64811E00
8.55961E00
7.47112E00
6.38263E00
5.29413E00
4.20564E00
3.11715E00
2.02865E00
9.40158E-01
1.48336E-01

(a) **(b)**



(c)

Figura 4.8. Distribución de flujos de calor para el ejemplo 3: (a) Algor, (b) Placa, (c) Cosmos Works.

En la figura 4.8 se puede apreciar la distribución de los flujos de calor a través del motor, los valores obtenidos tanto para Algor como para Placa son relativamente cercanos, con una variación de alrededor de 2 W/m^2. Por otro lado, los valores de Cosmos Works se alejan demasiado de los valores obtenidos en los otros dos programas, la posible causa de estas diferencias es el mallado distinto exhibido por Cosmos, el cual a diferencia de Placa y Algor es un mallado más estructurado, lo que ocasiona diferencias en los resultados. La figura 4.8 muestra singularidades en la distribución de los flujos alrededor de las fronteras de la geometría y estas son debidas, como se explicó anteriormente, a diferencias en el tipo de mallado de cada programa, pero se observa que existe uniformidad en la distribución para las zonas que no tienen especificadas condiciones de frontera, ya sea una temperatura o un flujo.

4.5. Ejemplo de verificación 4: Paredes compuestas expuestas a convección y generación de calor

Se tiene una pared compuesta de tres materiales distintos. La superficie exterior de la pared está expuesta a un fluido con un coeficiente de convección de $1000W/m^2K$ y a una temperatura de 25°C. La pared de la mitad experimenta una generación de calor de $4x10^6W/m^3$. Se busca encontrar la distribución de temperaturas en la pared. Las dimensiones de la pared se aprecian en la siguiente figura 4.9, así como sus propiedades térmicas.

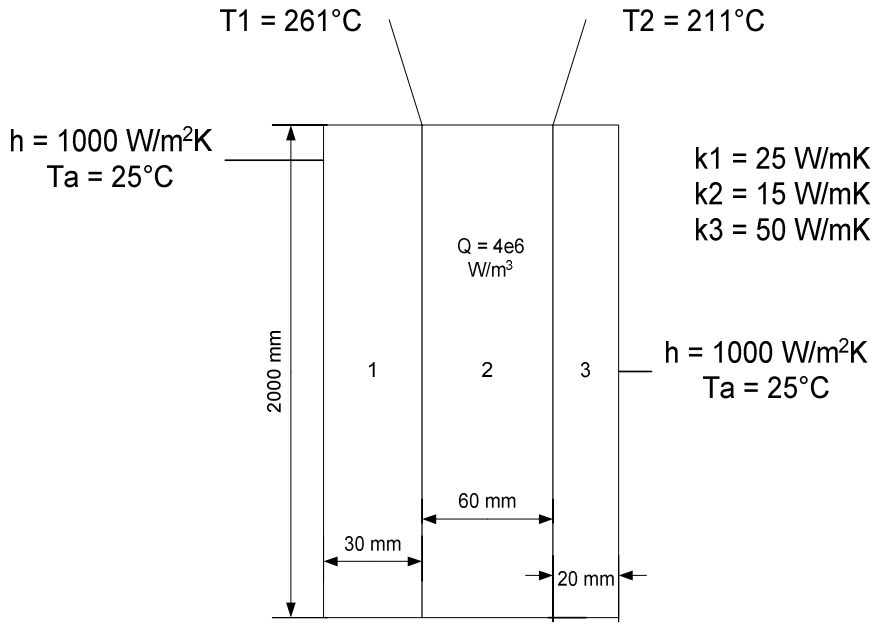


Figura 4.9. Geometría y condiciones de frontera para el ejemplo 4.

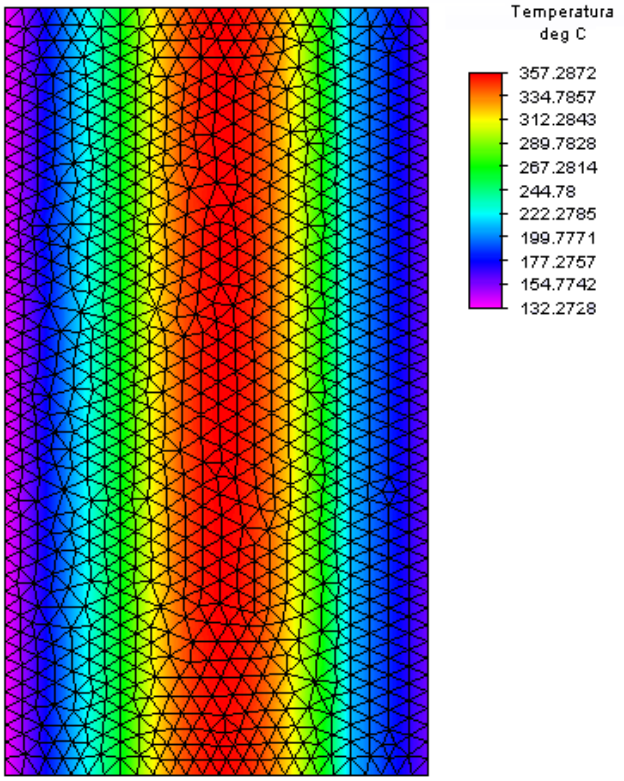
4.5.1. Comparación de resultados

La tabla 4.4 muestra los resultados obtenidos para mallas de aproximadamente 2000 elementos triangulares.

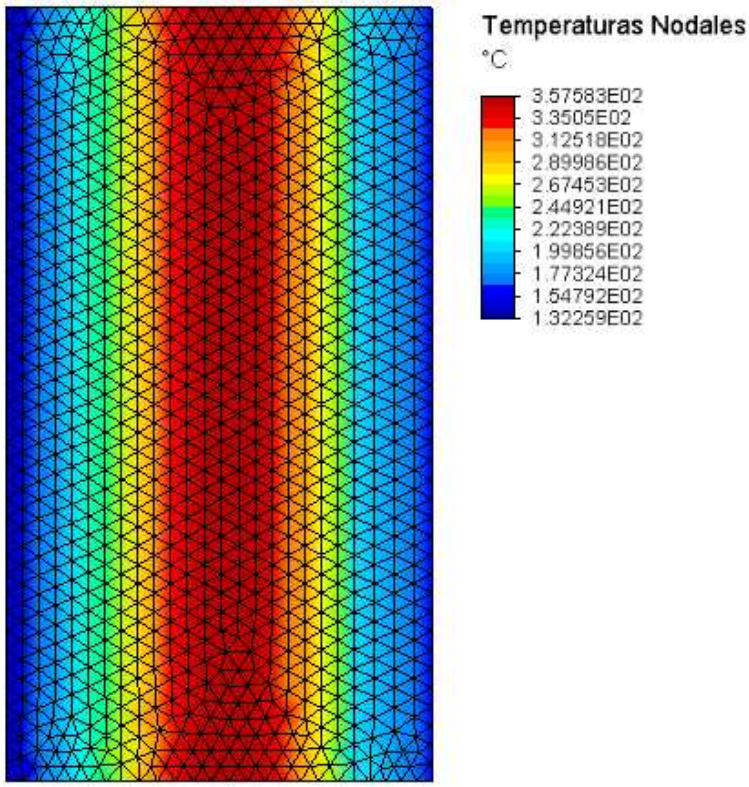
Tabla 4.4. Resultados obtenidos para el ejemplo 4.

Programas	**Algor**	**Placa**	**Cosmos Works**	**Solución Analítica[4]**
Forma de los elementos	Triangular	Triangular	Triangular	
Número de elementos	2088	2064	2028	
Temperatura máxima (°C)	357.287	357.58	357.2	358
Temperatura mínima (°C)	132.273	132.26	132.3	130

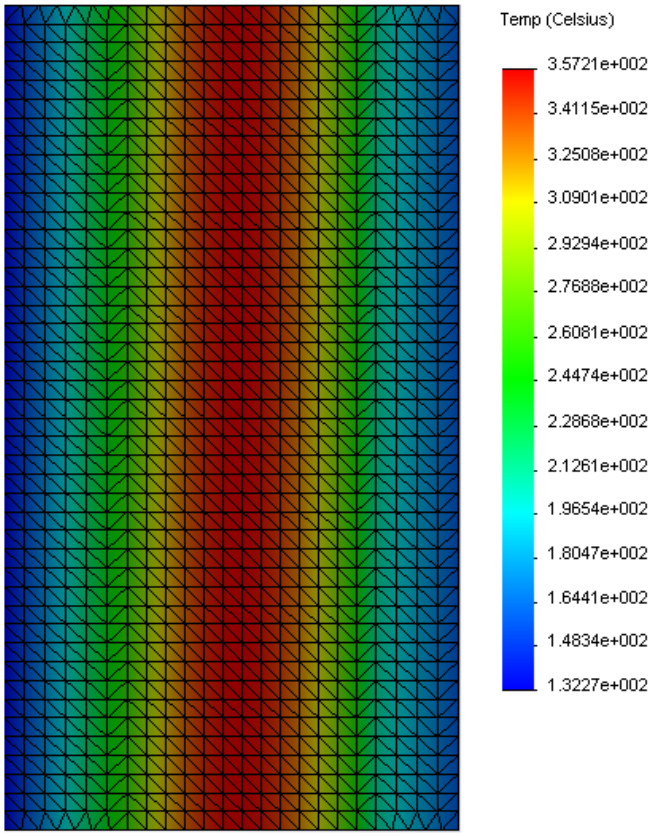
A continuación se presentan las distribuciones de temperatura obtenidas con aproximadamente 2000 elementos, con cada uno de los programas usados.


Temperatura
deg C
357.2872
334.7857
312.2843
289.7828
267.2814
244.78
222.2785
199.7771
177.2757
154.7742
132.2728

(a)


Temperaturas Nodales
°C
3.57583E02
3.3505E02
3.12518E02
2.89986E02
2.67453E02
2.44921E02
2.22389E02
1.99856E02
1.77324E02
1.54792E02
1.32259E02

(b)



(c)

Figura 4.10. Distribución de temperaturas para el ejemplo 4: (a) Algor, (b) Placa, (c) Cosmos Works.

Como se puede apreciar en la figura 4.10, las distribuciones de temperaturas obtenidas con los programas utilizados son similares y los valores obtenidos para cada nodo son prácticamente los mismos, y son muy similares con los valores obtenidos mediante la solución analítica. Por otro lado, los contornos apreciados en la gráfica de las distribuciones son similares para los tres programas.

4.6. Ejemplo de verificación 5: Turbina de gas

Se tiene una turbina de gas a la cual se le han maquinado unos canales rectangulares de 2 mm x 6 mm, para ofrecer un sistema de enfriamiento interno a las cuchillas de la turbina. Por estos canales rectangulares pasa aire a una temperatura de operación de 400K y un coeficiente de convección de 200 W/m^2K. La cuchilla de la turbina tiene 6 mm de ancho, 14 mm de largo y 300 mm de espesor, con una conductividad térmica de 25 W/mK. Las superficies superior e inferior de la cuchilla están expuestas a convección con los gases de combustión, los cuales se encuentran a 1700K y tienen un coeficiente de convección de 1000 W/m^2K. Se busca encontrar la distribución de temperaturas alrededor de la cuchilla de la turbina.

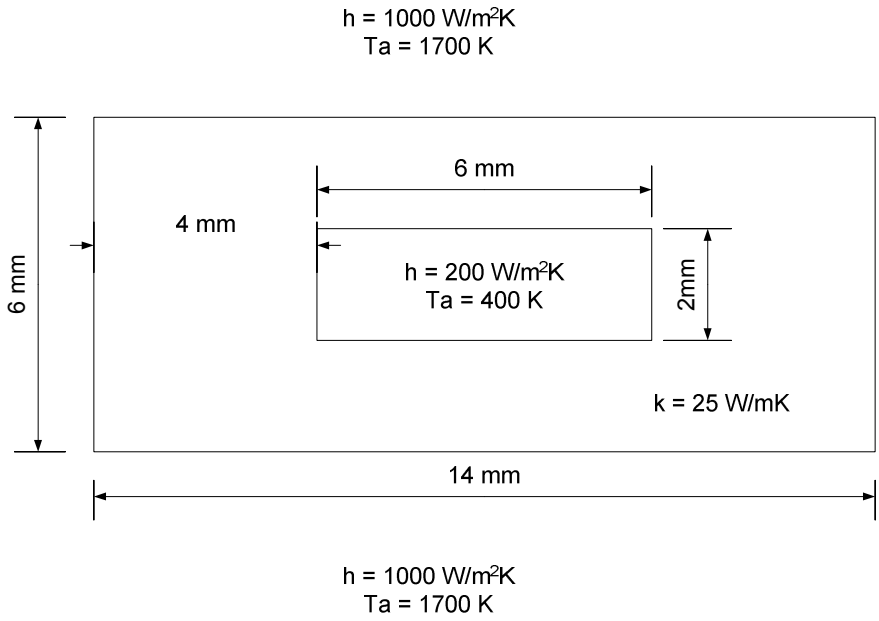


Figura 4.11. Condiciones del ejemplo 5.

4.6.1. Comparación de resultados

La tabla 4.5 muestra los resultados obtenidos para mallas de aproximadamente 300, 900 y 2500 elementos triangulares.

Tabla 4.5. Resultados obtenidos para el ejemplo 5.

Convergencia de Resultados Ejemplo 5			
Programa	**Algor**		
Forma de los elementos	Triangular		
Número de elementos	300	906	2536
Temperatura máxima (K)	1577.09	1577.1	1577.1
Temperatura mínima (K)	1545.2	1545.1	1545.1
Programa	**Placa**		
Forma de los elementos	Triangular		
Número de elementos	296	900	2500
Temperatura máxima (K)	1578.13	1577.1	1577.5
Temperatura mínima (K)	1544.8	1545.1	1544.9
Programa	**Cosmos Works**		
Forma de los elementos	Triangular		
Número de elementos	314	934	2534
Temperatura máxima (K)	1577	1577	1577
Temperatura mínima (K)	1545	1545	1545

A continuación se presentan las distribuciones de temperatura obtenidas con cada uno de los programas usados.

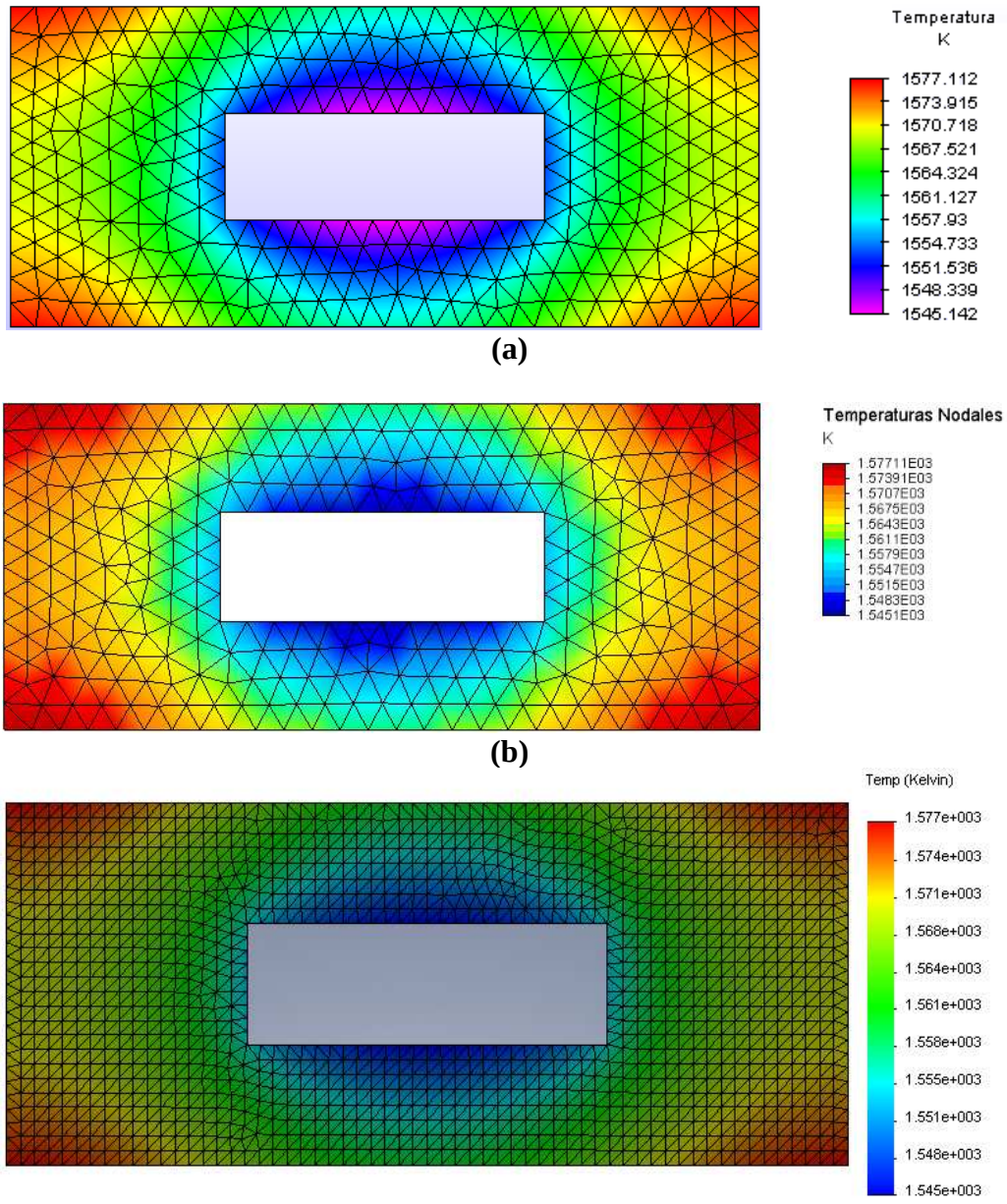


Figura 4.12. Distribución de temperaturas para el ejemplo 5: (a) Algor, (b) Placa, (c) Cosmos Works.

Como se puede apreciar en la figura anterior, las distribuciones de temperatura para Algor, Placa y Cosmos Works son similares y los valores obtenidos para cada nodo son muy parecidos, con una muy pequeña variación.

4.7. Ejemplo de verificación 6: Enfriamiento de circuitos integrados

Se tiene un sistema de enfriamiento para circuitos integrados, el cual se compone de una placa fría hecha de aluminio con una conductividad térmica de 190 W/mK, la cual tiene canales rectangulares regularmente distribuidos por donde pasa agua. Bajo condiciones normales existe un flujo de calor uniforme en la parte baja de la placa fría de 10E5 W/m^2; mientras que el agua se encuentra a un temperatura de 15°C y con un coeficiente de convección de 5000 W/m^2K. Utilizando la geometría que se presenta en la figura 4.13, se busca encontrar la distribución de temperaturas en la placa y la distribución del flujo de calor disipado en la placa fría.

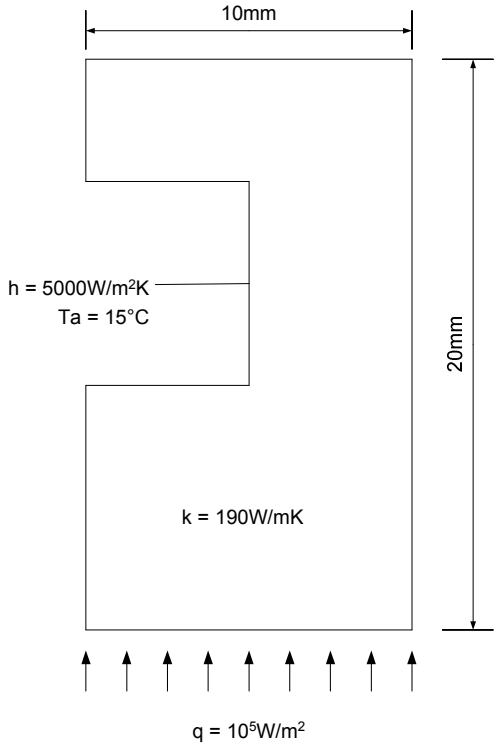


Figura 4.13. Geometría para el ejemplo 6.

4.7.1. Comparación de resultados

La tabla 4.6 muestra los resultados obtenidos para mallas de aproximadamente 1250, 2750 y 5000 elementos triangulares.

Tabla 4.6. Resultados obtenidos para el ejemplo 6.

Convergencia de Resultados Ejemplo 5				
Programa	Algor			**Solución Analítica[4]**
Forma de los elementos	Triangular			
Número de elementos	1247	2762	5034	
Temperatura máxima (K)	32.998	33.0054	33.0082	32.77
Temperatura mínima (K)	23.2359	23.231	23.2286	23.41
Programa	Placa			**Solución Analítica[4]**
Forma de los elementos	Triangular			
Número de elementos	1262	2526	4519	
Temperatura máxima (K)	32.999	33.088	33.2341	32.77
Temperatura mínima (K)	23.2357	23.222	23.2117	23.41

Programa	Cosmos Works			**Solución Analítica[4]**
Forma de los elementos	Triangular			
Número de elementos	1280	2707	5120	
Temperatura máxima (K)	32.86	32.86	32.86	32.77
Temperatura mínima (K)	23.08	23.08	23.08	23.41

A continuación se presentan las distribuciones de temperatura obtenidas con 2750 elementos, con cada uno de los programas usados.

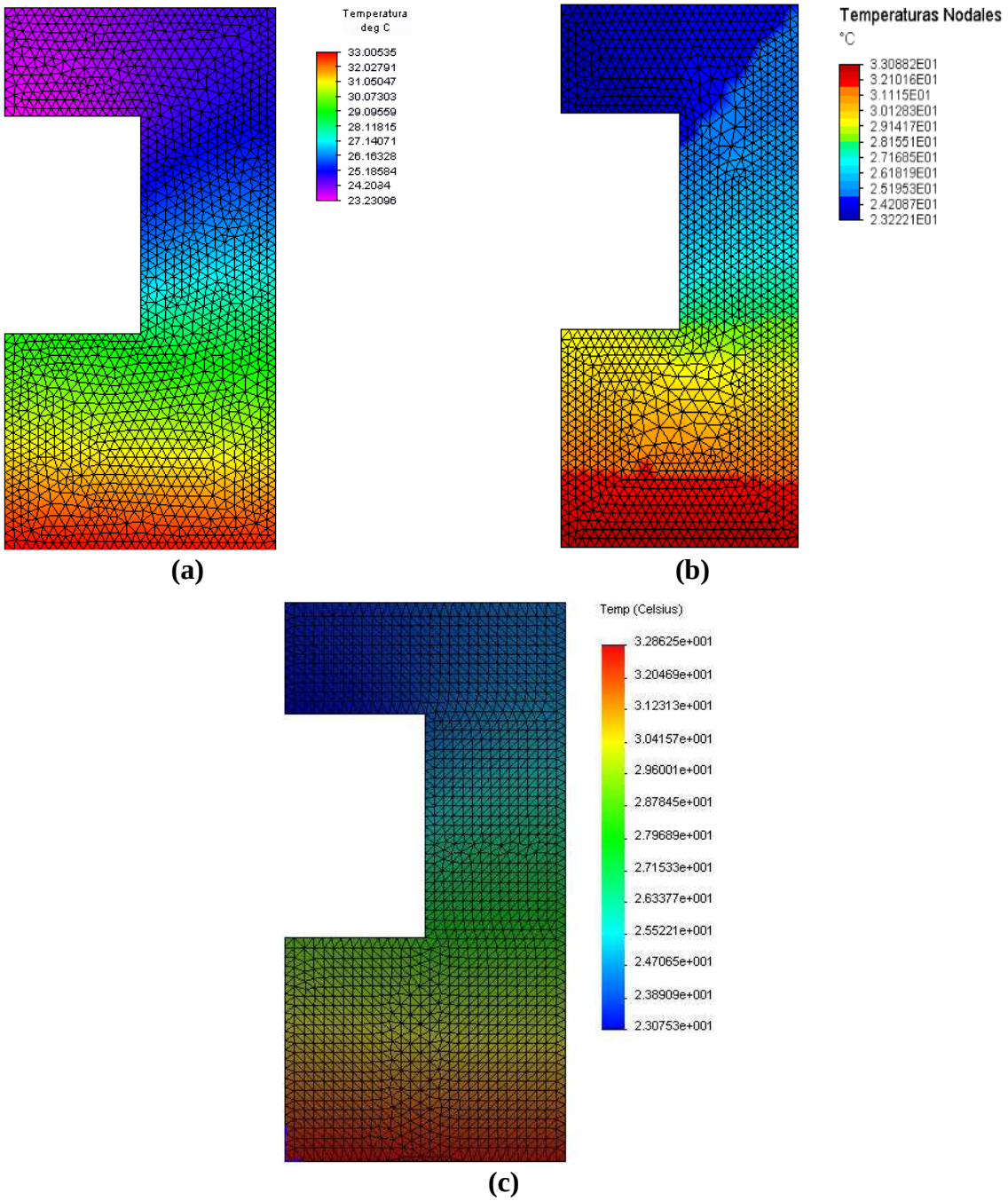


Figura 4.14. Distribución de temperaturas para el ejemplo 6: (a) Algor, (b) Placa, (c) Cosmos Works.

A continuación se muestran las distribuciones de los flujos de calor disipados con 2750 elementos, con cada uno de los programas.

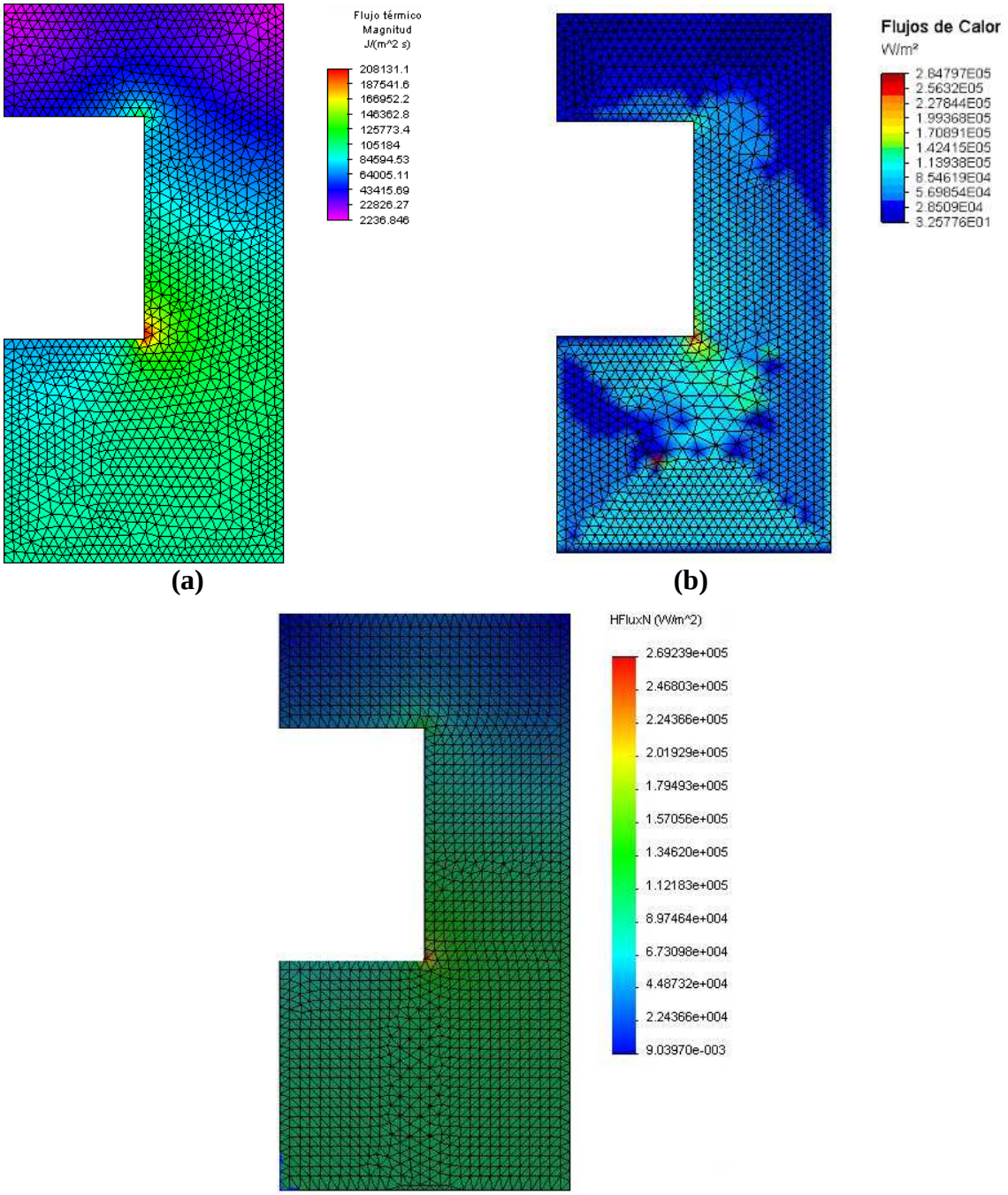


Figura 4.15. Distribución de los flujos de calor para el ejemplo 6: (a) Algor, (b) Placa, (c) Cosmos Works.

Como se observa en la figura 4.15, tanto Algor como Cosmos Works presentan una distribución de flujos bastante homogénea, mientras que Placa presenta una distribución no muy homogénea. En los resultados numéricos Placa presenta algunos picos y singularidades que se deben mayormente a la malla, ya que si comparamos las mallas obtenidas entre Algor y Placa se observan ciertas diferencias, una de las diferencias apreciadas es la homogeneidad en el tamaño de los elementos, la malla de Algor presenta elementos de tamaños similares mientras que Placa muestra algunos elementos que son muchos más grandes que otros y precisamente esos sectores de la malla provocan que se obtengan algunos resultados que distorsionan la distribución y que ocasionan resultados erróneos y que no son comparables entre los programas.

Por otra parte se observa, para los tres programas, que los valores de los flujos promedio tomando en cuenta los dos en los cuales no existe una condición de frontera se encuentran entre los 60000 y 80000 W/m^2. Estos valores, como se observa en la figura 4.15, están distribuidos de la siguiente manera: los valores más pequeños se encuentran en la parte superior de la geometría, con algunos valores grandes ubicados en las esquinas internas de la geometría. Estos valores son mayores en la parte central de la geometría y finalmente en la parte inferior, exceptuando la frontera donde se encuentra un flujo convectivo, están los valores más grandes.

4.8. Ejemplo de verificación 7: Placa sometida a distintas condiciones de temperatura

Se tiene una placa de 5x2x0.1 m^3, hecha de acero inoxidable AISI 202 con una conductividad térmica de 16.2 W/mK, a la cual se le somete a diferentes condiciones de temperatura como se aprecia en la figura 4.16. Se busca encontrar la distribución de temperaturas y la distribución de las tasas de flujo de calor.

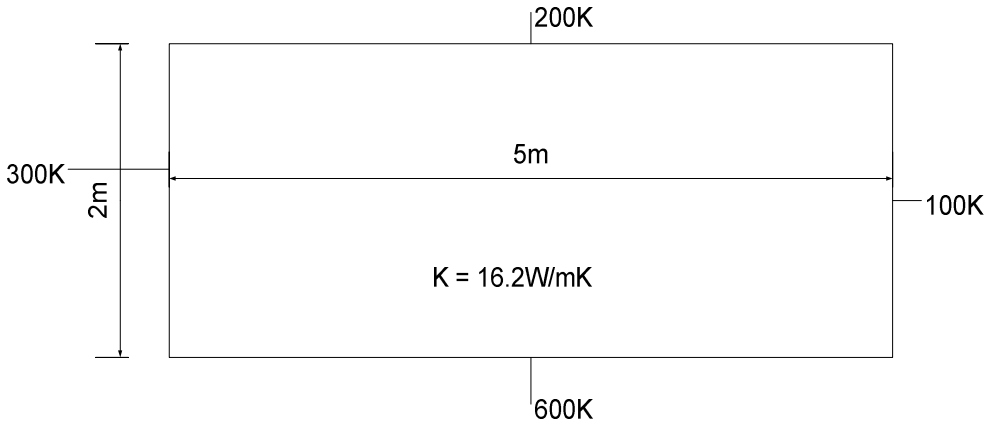


Figura 4.16: Dimensiones y condiciones ejemplo 7.

4.8.1. Comparación de resultados

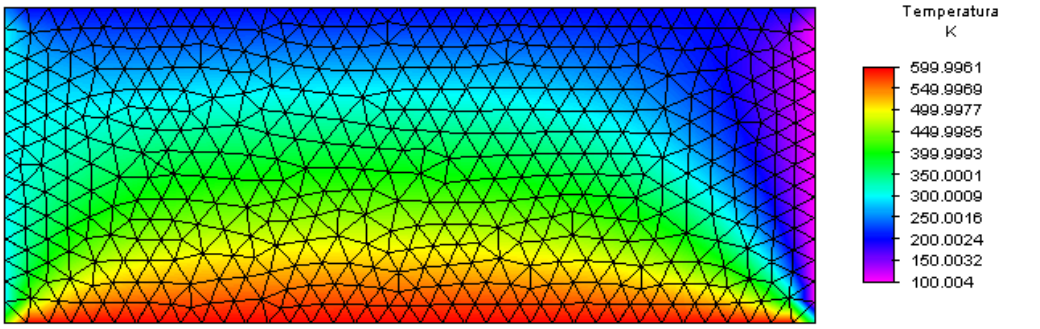
En este caso se compararan los resultados obtenidos usando dos tipos de elementos durante la discretización: elementos triangulares y elementos rectangulares. También se comparó con distintos números de elementos. En Cosmos Works solo se pueden modelar mallas estructuradas de elementos triangulares.

Tabla 4.7. Resultados obtenidos para el ejemplo 7 utilizando elementos triangulares.

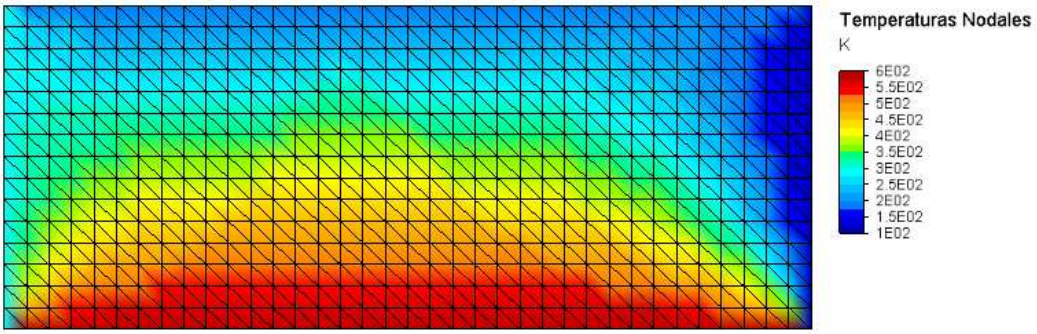
Convergencia de resultados en el ejemplo 7			
Programa	**Algor**		
Forma de los elementos	Triangular		
Número de elementos	222	1088	2518
Temperatura máxima (K)	600	600	600
Temperatura mínima (K)	100	100	100
Flujo de calor promedio en la placa (W/m^2)	**3668.43**		
Programa	**Placa**		
Forma de los elementos	Triangular		
Número de elementos	224	1080	2576
Temperatura máxima (K)	600	600	600

Temperatura mínima (K)	100	100	100
Flujo de calor promedio en la placa (W/m^2)	**3538.69**		
Programa	**Cosmos Works**		
Forma de los elementos	Triangular		
Número de elementos	226	1082	2578
Temperatura máxima (K)	600	600	600
Temperatura mínima (K)	100	100	100
Flujo de calor promedio en la placa (W/m^2)	**3963.33**		

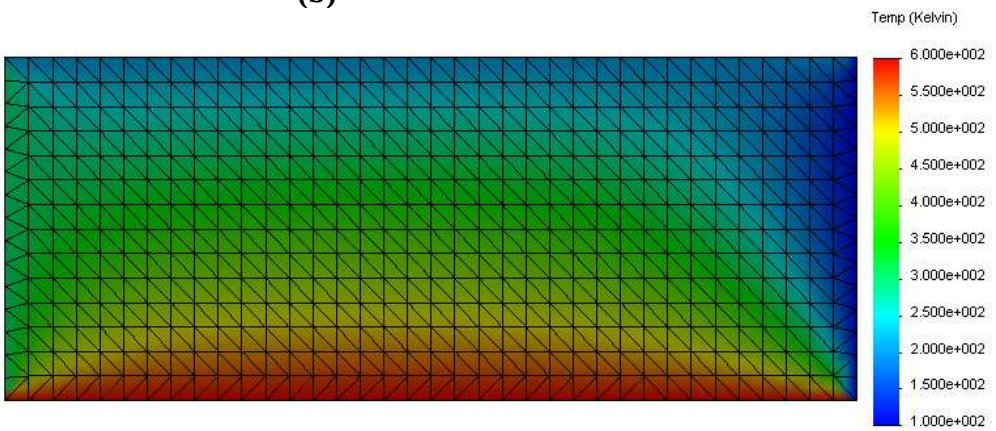
Las figuras 4.17 y 4.18 muestran la distribución de temperaturas y la distribución de las tasas de flujo de calor para mallados con 1000 elementos triangulares.



(a)



(b)



(c)

Figura 4.17. Distribución de temperaturas para el ejemplo 7: (a) Algor, (b) Placa, (c) Cosmos Works.

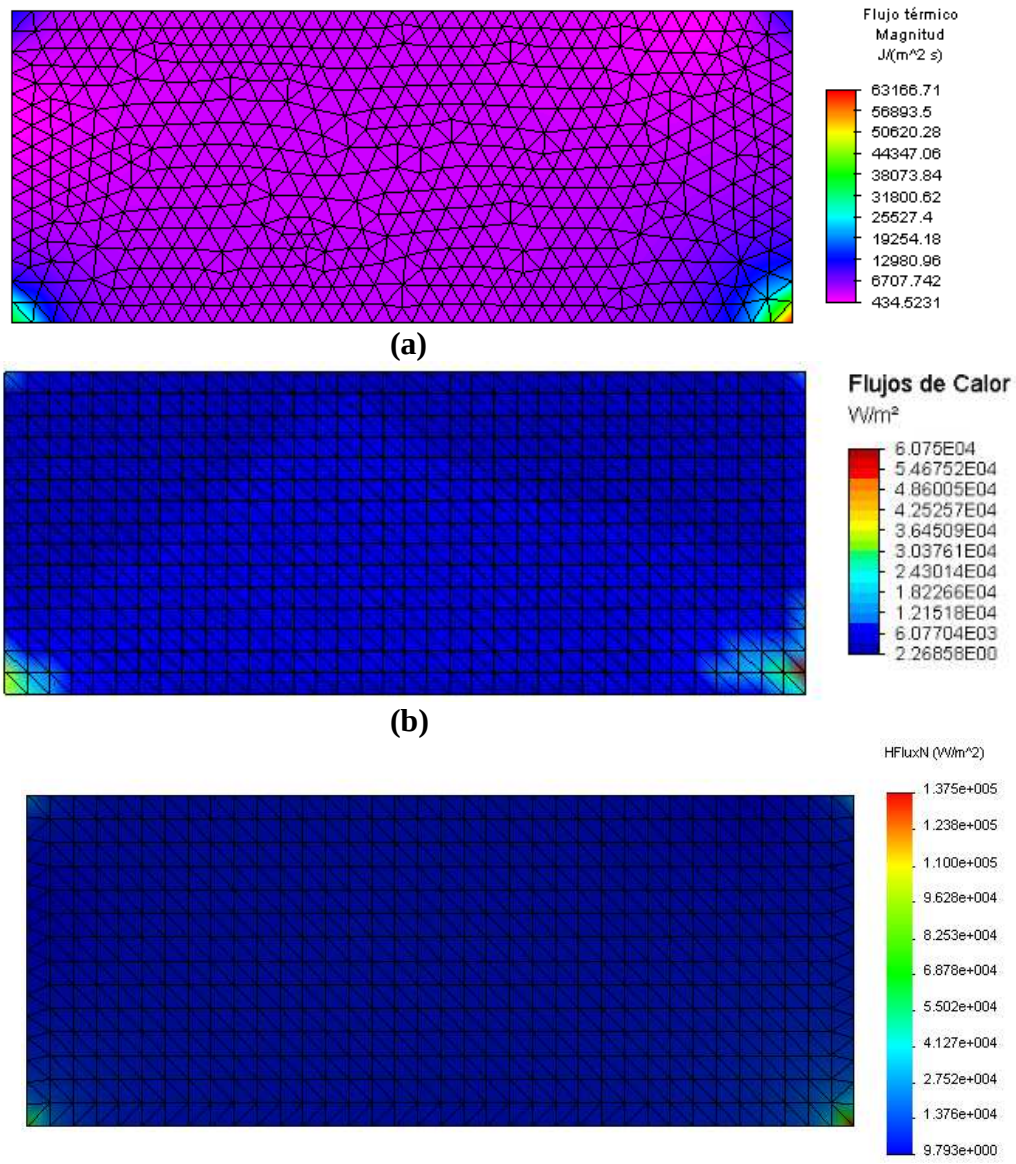


Figura 4.18. Distribución de la tasa de calor para el ejemplo 7: (a) Algor, (b) Placa, (c) Cosmos Works.

Las distribuciones observadas en la figura 4.18 presentan diferencias en los valores máximos debido a que las mallas presentadas para cada programa difieren una de la otra. Tanto Placa como Cosmos Works presentan mallas más estructuradas, mientras que Algor presenta una malla con elementos dispuestos de manera aleatoria. Estos valores máximos son valores puntuales y no representan lo que ocurre en realidad con la distribución de los flujos.

Se puede inferir dado que la diferencia de temperaturas entre el borde inferior y el borde derecho de la placa es la más grande, se presenta un mayor flujo entre esos dos bordes, en cambio que desde el borde superior al borde izquierdo y derecho existe la misma diferencia de temperaturas por lo que la distribución entre esos dos bordes de la placa es similar.

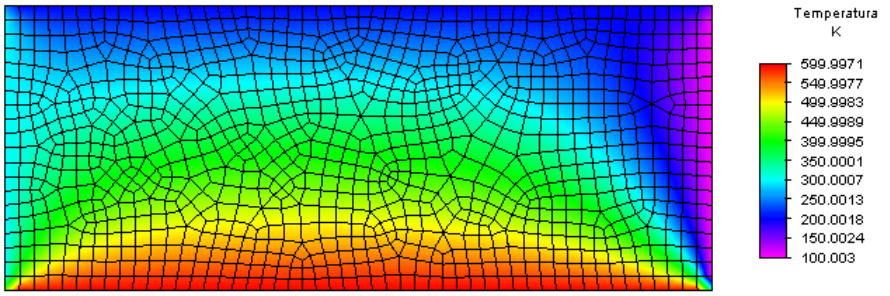
El valor promedio de la tasa de calor se calculó tomando los valores de los nodos que no se encuentran en las fronteras del dominio, en Cosmos Works y Algor este valor se lo

obtiene automáticamente escogiendo los nodos requeridos y eligiendo la opción de promedio, mientras que en Placa se hizo el cálculo de forma manual, dichos valores se encuentran en la Tabla 4.7 y como se puede apreciar están entre los 3500 y 4000 W/m^2. Se puede concluir que los valores de tasas de calor en la placa son mayores en la parte baja de la geometría y menores en la parte superior de la misma, con ciertas singularidades dadas por los valores puntuales y por las condiciones de las mallas.

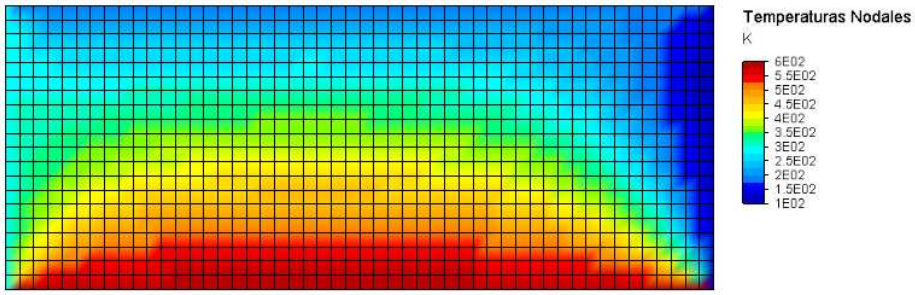
Tabla 4.8. Resultados obtenidos para el ejemplo 7 utilizando elementos rectangulares.

Convergencia de resultados en el ejemplo 7						
Programa	**Algor**			**Placa**		
Forma de los elementos	Cuadrangular			Rectangular		
Número de elementos	234	998	2592	207	1000	2560
Temperatura máxima (K)	600	600	600	600	600	600
Temperatura mínima (K)	100	100	100	100	100	100
Flujo de calor promedio de la placa (W/m^2)	3712.3			4025.92		

Las figuras 4.19 y 4.20 representan la distribución de temperaturas y la distribución de las tasas de calor para 1000 elementos rectangulares.



(a)



(b)

Figura 4.19. Distribución de temperaturas ejemplo 7: (a) Algor, (b) Placa.

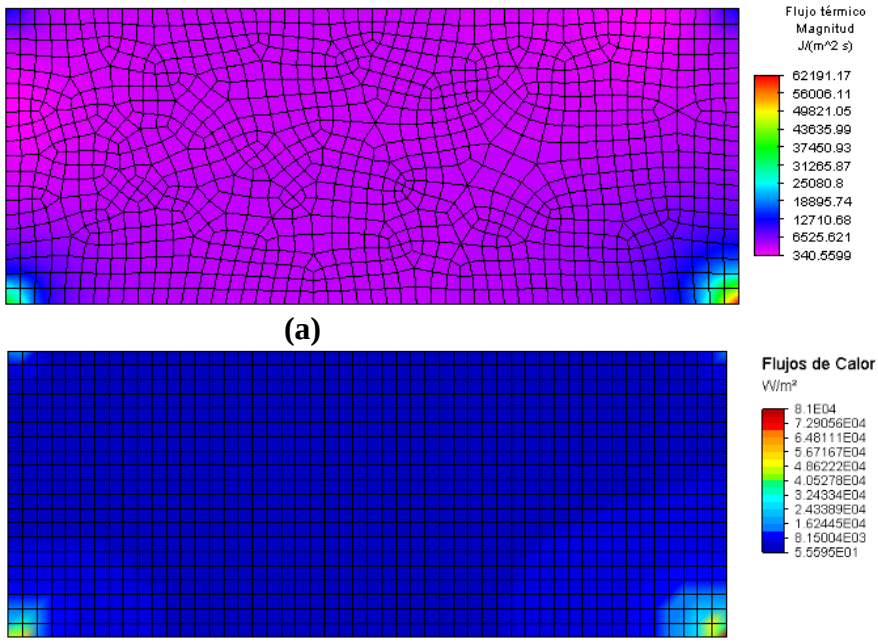


Figura 4.20. Distribución de la tasa de calor para el ejemplo 7: (a) Algor, (b) Placa.

En la figura 4.20 se observa una mayor uniformidad entre las distribuciones obtenidas tanto para Placa como para Algor, a pesar de que la malla de elementos cuadrangulares producida por Algor no tenga la misma homogeneidad como la presentada por Placa. Nuevamente se aprecia que los valores máximos obtenidos son frutos de estas diferencias y son valores puntuales que representan la mayor diferencia que existe entre el borde inferior y el borde derecho de la placa.

Por esta razón los valores de la tasa de calor en la placa se los obtuvo haciendo un promedio de los valores de los nodos que no se encuentran en las fronteras, dando como resultado los valores expresados en la Tabla 4.8, dichos valores se encuentran entre los 3700 y los 4000 W/m^2.

Es importante también recalcar la diferencia que existe entre los valores promedio obtenidos para mallas triangulares y para mallas rectangulares. Como se puede apreciar en las Tablas 4.7 y 4.8 los valores obtenidos para Algor en los dos tipos de malla son bastante aproximados, mientras que en Placa se observa una diferencia de un poco más del 10% entre el resultado obtenido con mallas de elementos triangulares y el resultado con mallas de elementos rectangulares, esa diferencia es debida al número de nodos tomados para calcular los resultados, ya que en el caso de elementos rectangulares el número de nodos tomados es casi el doble de los tomados para elementos triangulares.

4.9 Comparación de tiempos de mallado y de análisis

Una vez comprobado el funcionamiento del programa Placa, es importante también hacer una comparación del tiempo requerido por el programa para realizar las dos tareas más importantes del proceso de simulación mediante elementos finitos: el mallado de la geometría dibujada y el análisis propiamente dicho. Para comparar los tiempos que los tres programas emplean para mallar y hacer el análisis se escogió la geometría y las condiciones de frontera del problema de verificación 1 de la sección anterior.

Como se dijo anteriormente, Placa utiliza la técnica de avance frontal para realizar los mallados triangulares no estructurados. Además, Placa tiene la posibilidad de hacer mallas estructuradas utilizando elementos triangulares y rectangulares, pero solo para geometrías rectangulares. Por otra parte, Algor produce solo mallas no estructuradas de elementos triangulares, cuadriláteros o de combinación entre estos dos elementos. Finalmente, Cosmos Works solo permite la creación de mallas estructuradas con elementos triangulares.

En la figura 4.21 se muestran los tiempos empleados en cada programa para producir mallas triangulares no estructuradas para el caso de Algor y Placa, y mallas triangulares estructuradas para Cosmos Works. Las curvas de tiempos están en función del número de elementos.

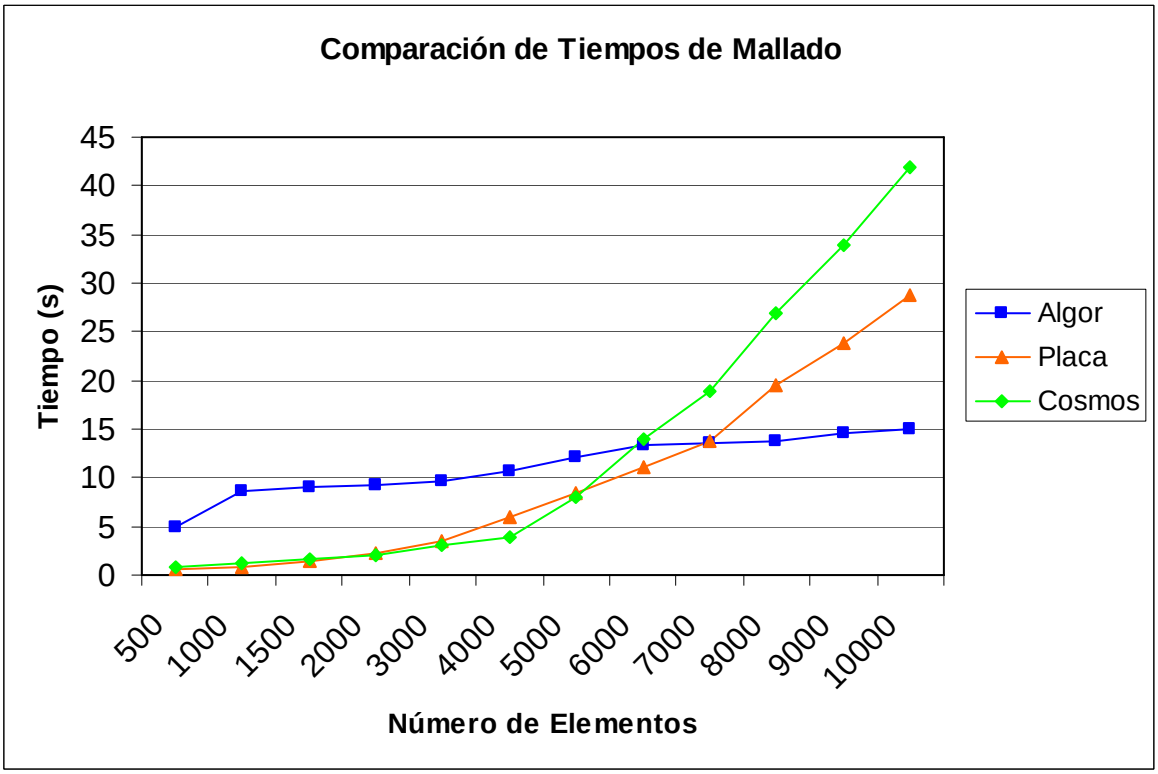


Figura 4.21. Comparación de tiempos de mallado.

Como se puede apreciar en la figura 4.21, para el problema seleccionado, Algor presenta unos tiempos de mallado menores a los otros dos programas en mallas superiores a los 6000 elementos, mientras que en mallas menores a 6000 elementos tanto Placa como Cosmos Works presentan unos tiempos similares, aunque al sobrepasar los 6000 elementos Cosmos Works es el programa menos eficiente en los tiempos del mallado.

Por otra parte, para comparar los tiempos empleados en el análisis se procedió de igual manera que para los tiempos del mallado, se cambió las densidades de las mallas desde 500 elementos hasta los 10000 elementos y se observó el tiempo que se demoraban en realizar el análisis tomando como condiciones de frontera las utilizadas en el ejemplo del literal 4.2. La figura 4.22 muestra los distintos tiempos para los tres programas utilizados.

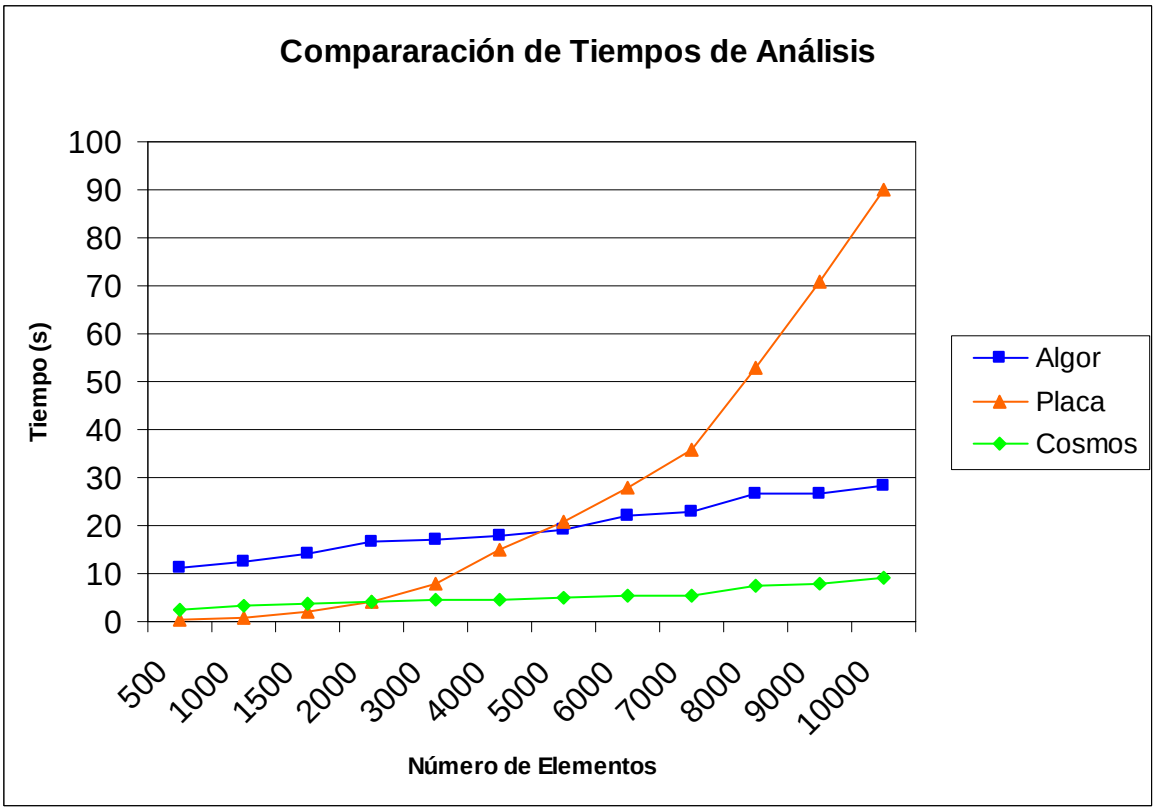


Figura 4.22. Comparación de tiempos de análisis.

Se puede apreciar en la figura anterior que el programa más eficiente y más rápido en realizar el análisis para el distinto número de elementos resultó ser Cosmos Works, seguido de Algor y finalmente el menos eficiente fue Placa. En este programa se observa que a medida que hay más elementos, la pendiente de la curva de tiempo de análisis crece rápidamente, no así en los otros dos programas utilizados.

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

El presente trabajo, junto con el programa Placa, demuestran que es posible desarrollar software para la resolución de problemas de ingeniería con la utilización de un método matemático implementado de forma computacional y que no requiere de una inversión extremadamente alta tanto para el desarrollador como para el consumidor. La

utilización de lenguajes de programación que no requieren una gran destreza y un amplio conocimiento del mismo al momento de programar es otro de los logros que se han conseguido con Placa y que es importante de recalcar.

Con respecto a las pruebas comparativas realizadas entre el programa Placa, Algor y Cosmos Works, se puede concluir a partir de los resultados obtenidos que el programa funciona correctamente y que entrega resultados que son coherentes y que son bastante similares a los obtenidos con los programas comerciales anteriormente mencionados y las soluciones analíticas presentadas en algunos de los ejemplos. Se observa también que debido a las distintas construcciones y arquitecturas de los programas, existen ciertas diferencias en algunos resultados pero que son producto de las diferencias en la estructuración y en la forma de calcular los resultados que tiene cada programa, factor que no influye en la conclusión de que el programa desarrollado es eficiente y realiza el análisis de transferencia de calor de forma correcta.

En la comparación para los tiempos de mallado, se observó que Algor es el programa más eficiente al momento de producir mallas con una gran cantidad de elementos en un tiempo menor. Por otra parte, tanto Placa como Cosmos Works son eficientes al mallar mallas con un número menor de elementos pero a medida que el número aumenta el tiempo de mallado también, produciendo una demora mayor en el mallado de las geometrías. Sin embargo, el hecho de que el programa Placa pueda crear mallas de alrededor de 10000 elementos en menos de 30 segundos es un indicativo de que es posible utilizar el programa para producir mallas más densas de buena calidad en un tiempo razonablemente rápido.

Por otra parte, al momento de comparar los tiempos de análisis entre los programas se concluyó que el programa Placa es el más lento de los tres cuando las mallas de los problemas son bastante densas. Esto representa un limitante si se desea realizar análisis que impliquen tener mallas con más de 10000 elementos, pero estos 2 minutos que se demora el programa en realizar el análisis comparado con el tiempo que requeriría realizar el mismo análisis siguiendo una forma analítica, es relativamente insignificante.

Finalmente, como se dijo al principio del trabajo, esta versión del programa presentado no es la primera ni la última. Lo que se buscó desde en un principio es crear un programa base que sirva para ir aumentando y modificando las potencialidades del mismo, para poder ir mejorándolo, ya sea en su forma estética, en los procedimientos de análisis, optar por nuevos métodos de mallado, etc. Es decir, el programa fue creado con la finalidad de que siga creciendo y expandiéndose logrando así que se pueda en un mediano plazo tener un software completo que incluya varios tipos de análisis de ingeniería. Con esto se lograría que pueda ser utilizado por los propios estudiantes y porque no por los maestros al momento de enseñar ya sea un curso de Análisis Numérico o un curso de Elementos Finitos y que sirva como una herramienta en la enseñanza y en el aprendizaje de estas y otras materias de cualquier rama de la Ingeniería.

5.2 Recomendaciones para futuros trabajos

Como se dijo anteriormente, este programa ha sido creado con la finalidad de que pueda convertirse en una herramienta completa para la resolución de problemas de ingeniería usando el método de los elementos finitos. Por esta razón, el programa presentado se lo diseñó de forma tal que se puedan implementar nuevas opciones y

mejoras, que lograrían aumentar las potencialidades del mismo. Algunas de las características del programa que se pueden mejorar e implementar en trabajos futuros son las siguientes:

Mejorar el mallador.- Actualmente el programa Placa presenta dos tipos de malla, la estructurada y la no estructurada. Para la malla no estructurada el programa utiliza el método de avance frontal. Al no ser este el único método que se puede emplear al momento de crear mallas, sería excelente que se pueda implementar otro método de mallado que permita obtener mejores mallas, mejorando de esta forma el análisis de los problemas planteados.

Implementar resolutores propios.- Placa utiliza librerías de distribución gratuita para la resolución matricial de los problemas. Si se desarrollasen algoritmos de cálculos específicos, el programa se demoraría un menor tiempo en realizar el análisis y reduciría el tamaño y la memoria utilizada por el mismo.

Tipos de análisis y elementos.- Al aumentar los tipos de análisis y los tipos de elementos, el programa ya no estaría limitado a dos tipos de análisis y algunos tipos de elementos sino que la variedad y la gama de posibilidades que presentaría, permitirían que se considere a este software como una herramienta de gran utilidad para resolver problemas de ingeniería con el método de los elementos finitos.

REFERENCIAS

[1] Friedel, David Jr. y Potts Anthony. *Java Programming Language Handbook*. Scottsdale: Coriolis Group, 1996.

[2] Hunter, Peter. *Finite Element Method and Boundary Element Method.* Auckland: University of Auckland, 2003.

[3] Hutton, David. *Fundamentals of Finite Element Analysis*. New York: McGraw Hill, 2004.

[4] Incropera, Frank y DeWitt David. *Introduction to Heat Transfer*. 4ed. New York: John Wiley & Sons, 2002.

[5] Lewis, Roland; Nithiarasu, Perumal y Seetharamu, Kankanhalli. *Fundamentals of the Finite Element Method for Heat and Fluid Flow*. Londres: John Wiley & Sons, 2004.

[6] Manrique, José. *Transferencia de Calor*. México D.F.: Harla, 1976.

[7] Mills, Anthony. *Heat Transfer*. Concord: Irwin, 1992.

[8] Moaveni, Sabed. *Finite Element Analysis: Theory and Application with ANSYS*. Nueva Jersey: Prentice Hall, 1999.

[9] Nicholson, David. *Finite Element Analysis: Thermomechanics of Solids*. Boca Raton: CRC Press, 2003.

[10] Oñate, Eugenio y Zárate Francisco. *XII Curso de Máster en Métodos Numéricos para Cálculos y Diseño en Ingeniería: Introducción al Método de los Elementos Finitos.* 2001.

[11] Ozisik, Necati. *Finite Difference Method in Heat Transfer*. Boca Raton: CRC Press, 1994.

[12] Romero, Eduardo. *Tesis: Desarrollo de Software para el Análisis Bidimensional de Placas mediante el Método de los Elementos Finitos*. Quito: Escuela Politécnica Nacional, 2006.

[13] Romero, Eduardo. *El Lenguaje de Programación Java*. Quito: Escuela Politécnica Nacional, 2006.

[14] Sánchez, Julio y Canton, María. *Java Programming for Engineers.* Boca Raton: CRC Press, 2002.

[15] Zienkiewicz, O.C. *El método de los elementos finitos*. Barcelona: Reverté, 1982.

[16] Yoshimoto, S. Nakasone, Y. y Stolarski, T. *Engineering Analysis with ANSYS Software*. Oxford: Elsevier, 2006.

[17] http://www.algor.com/products/productsheets/corporate_flyer.pdf

[18] http://www.solidworks.es/pages/products/cosmos/cosmosworks.html