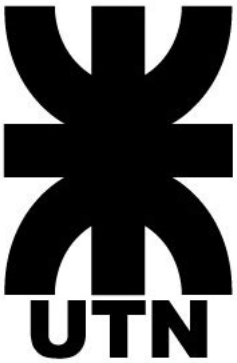
MATEMÁTICA SUPERIOR

ECUACIÓN DEL CALOR EN 2D



Facultad regional Concepción del Uruguay

Alumno: Leonardo Alfonso Hoet

Profesores:
Dr. Omar Roberto Faure
Ing. Gaston Maximiliano Quiroga

Carrera: Ingeniería en sistemas de información

Fecha de entrega: 11/02/2020

Índice

1. Introducción **3**

2. Modelo matemático **4**
2.1. Consideraciones 4
2.2. Desarrollo 5
2.3. Ejemplo 9
2.4. Análisis del error 10

3. Implementación **12**
3.1. Imágenes 12
3.2. Código 14

4. Conclusión **15**

5. Bibliografía **16**
5.1. Apuntes 16
5.2. Software 16

Anexo **17**

A. Cógido en octave **17**

1. Introducción

Con la llegada de procesadores más potentes se ha incrementado tanto el poder de computo como la energía consumida por estos. Debido a que los procesadores son objetos físicos no ideales, cuando funcionan con el paso de la corriente generan calor. Esto puede ser un gran inconveniente, ya que con el aumento de la temperatura se dilatan las finas pistas de electrónica hasta el punto de llegar a hacer corto circuito y dejar de funcionar.Es por esto, que cualquier computadora moderna tiene incorporado disipadores, generalmente de Aluminio o Cobre

En el siguiente trabajo, nos proponemos desarrollar un modelo matemático bidimensional para el estudio de la disipación de energía en estos disipadores, estudiar y predecir la variación temperatura con respecto al tiempo, como así también la aplicación del mismo. Buscaremos una solución numérica al problema, ya que en la ingeniería muchos fenómenos están determinados por ecuaciones diferenciales las cuales no se puede hallar una solución simbólica.

2. Modelo matemático

2.1. Consideraciones

Para comenzar, asumiremos los siguientes enunciados:

Aproximación de la derivada primera Sea f una función de $U \subseteq \mathbb{R} \to \mathbb{R}$, de al menos clase ζ^1. Sea $x_0 \in U$. Se considera una buena aproximación de la primera derivada de f a la formula

$$\frac{df(x_0)}{dx} \approx \frac{f(x_0 + h) - f(x_0)}{h} \tag{1}$$

si h es lo suficientemente pequeña.

Aproximación de la derivada segunda Sea f una función de $U \subseteq \mathbb{R} \to \mathbb{R}$, de al menos clase ζ^2. Sea $x_0 \in U$. Se considera una buena aproximación de la segunda derivada de f a la formula

$$\frac{d^2f(x_0)}{dx^2} \approx \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{2h} \tag{2}$$

si h es lo suficientemente pequeña.

Evolución de la temperatura en el tiempo Sea $u : U \subseteq \mathbb{R}^3 \to \mathbb{R}$ una función de al menos clase ζ^2. Esta función toma como argumentos (x, y, t) y nos devuelve una determinada temperatura para ese punto en un determinado tiempo. La evolución de la temperatura en un sistema físico esta dada por:

$$\frac{\partial u}{\partial t} = c\,(\nabla^2 u) \tag{3}$$

donde

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

y c es una constante física que depende del material

2.2. Desarrollo

Para comenzar, reemplazaremos las derivadas[1] de la fórmula 3 por la fórmula 1 y la fórmula 2

$$\frac{\partial u}{\partial t} = c\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \approx \frac{u_{i,j,k+\Delta t} - u_{i,j,k}}{\Delta t} =$$

$$= c\left(\frac{u_{i+\Delta x,j,k} - 2u_{i,j,k} + u_{i-\Delta x,j,k}}{(\Delta x)^2} + \frac{u_{i,j+\Delta y,k} - 2u_{i,j,k} + u_{i,j-\Delta y,k}}{(\Delta y)^2}\right)$$

Podemos observar que los puntos cuya posición temporal es k son aquellos puntos en el presente, los cuales conocemos su temperatura. Procedemos a despejar $u_{i,j,k+1}$, es decir el punto del que queremos predecir la temperatura. Luego de despejar y un desarrollo algebraico, obtenemos:

$$u_{i,j,k+1} = (1-4r)u_{i,j,k} + r(u_{i-\Delta x,j,k} + u_{i+\Delta x,j,k} + u_{i,j+\Delta y,k} + u_{i,j-\Delta y,k}) \quad (4)$$

Siendo $\Delta y = \Delta x$ y $r = \frac{\Delta t\, k}{\rho\, c\, (\Delta x)^2}$

Observando la fórmula, se puede deducir que la temperatura de un punto en el instante de tiempo siguiente depende de la temperatura en ese mismo punto y la temperatura de los puntos adyacentes. Esto tiene sentido, ya que las aproximaciones 1 y 2 nos están diciendo que para conocer la tasa de cambio de la función en un punto, debemos mirar ese mismo punto y los adyacentes.

Como tenemos una placa bidimensional y estamos trabajo con diferencias finitas, podemos enumerar los nodos sobre los cuales haremos nuestros cálculos de manera tal que necesitemos solo una dimensión para representarlos. Nuestro criterio de ordenamiento sera de izquierda a derecha y de abajo hacia arriba. Si tenemos una placa rectangular, el punto que este mas abajo y mas a la izquierda será el numero 1, el inmediato hacia la derecha será 2 y así sucesivamente. Cuando se llegue a un borde se seguirá con la misma numeración en la fila superior. Por este criterio y como la placa es rectangular, obtenemos una n cantidad de puntos en una fila y m cantidad de filas. Esto se puede observar de una manera mas clara en la figura 1

Este criterio de ordenamiento y la fórmula 4 nos permite plantear el problema como un sistema lineal de la manera:

$$Mu^k = u^{k+1}$$

[1]Para simplificar la notación usaremos $u(x_i, y_j, t_k) = u_{i,j,k}$

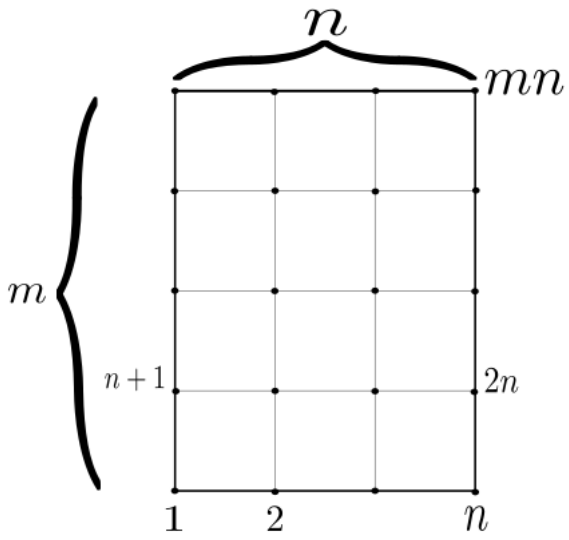


Figura 1: Numeración de los nodos

donde u^k es un vector que representa los puntos en un instante de tiempo determinado y M una matriz.

Para identificar cada nodo en el vector u se seguirá el siguiente método: Dado un punto u_j se puede determinar su inmediato hacia la derecha como u_{j+1} y su inmediato superior como u_{j+n}. Análogamente se procederá para los puntos inferiores o izquierdos. Con este sistema podemos reescribir 4 como:

$$u_{j,k+1} = (1 - 4r)u_j + r(u_{j-n,k} + u_{j+n,k} + u_{j-1,k} + u_{j+1,k}) \tag{5}$$

Esto nos permite implementar la matriz M como una matriz de $n * m$ filas por $n * m$ columnas, donde en el sistema lineal anteriormente planteado, el elemento $a_{i,i}$ de dicha matriz multiplica a un punto u_j; $a_{i-n,i}$ multiplica a u_{j-n} y $a_{i-1,i}$ multiplica a u_{j-1}.

Condiciones de borde Si consideramos que la placa esta hecha con un material homogéneo, podemos calcular solo la mitad de ella sin perder datos, ya que para tener los datos completos solo habría que espejar nuestro proceso y rehacer los cálculos. De esta manera, las siguientes condiciones de borde se plantearan teniendo en cuenta que solo analizamos que pasa en la mitad superior de la placa.

Para tener una solución única al problema necesitamos tener condiciones de borde; en particular 4 de ellas, que corresponden a cada borde de nuestra

placa. Nuestro modelo nos dice que pasa con la temperatura de los puntos en el interior de la placa, pero no nos dice nada acerca de que pasa en los extremos. Para ello planteamos cuatro condiciones

I) Condición inicial Los puntos por donde ingresa calor, en este caso en la recta $x = 0$, para todo t e y la temperatura esta dada por una función $f : U \subseteq \mathbb{R}^2 \to \mathbb{R}$, es decir $u(0, y, t) = f(y, t)$

II) Condición de perdida en el extremo En la mayoría de los disipadores comerciales, la longitud es varios ordenes de magnitud mas grande que el ancho del mismo. Por esto las perdidas de calor en el extremo (cuando $x = L$) son despreciables, es decir:

$$\frac{\partial u}{\partial \eta}(L, y, t) = 0$$

III) Condición de la recta longitudinal inferior En nuestro caso de estudio solo consideramos la mitad superior del disipador. Consideramos que no hay transferencia de calor entre los puntos de la mitad superior y los puntos en la mitad inferior; cuando $y = 0$.

$$\frac{\partial u}{\partial \eta}(x, 0, t) = 0$$

IV) Condición de perdida por el aire Consideramos que todo la energía que disipa la placa lo hace mediante conducción de calor hacia el aire que tiene a su alrededor.

$$\frac{\partial u}{\partial \eta} - au = 0$$

Si ampliamos esta formula con diferencias finitas, llegamos a que

$$u_{j^*} = \frac{h_c}{k} 2h \; u_j + u_{j-n}$$

donde u_j es el punto en la frontera y u_{j^*} es un punto de aire. Con esto podremos calcular los puntos en la frontera si calculamos los puntos de aire.

Estas condiciones se ven reflejadas de la siguientes manera:

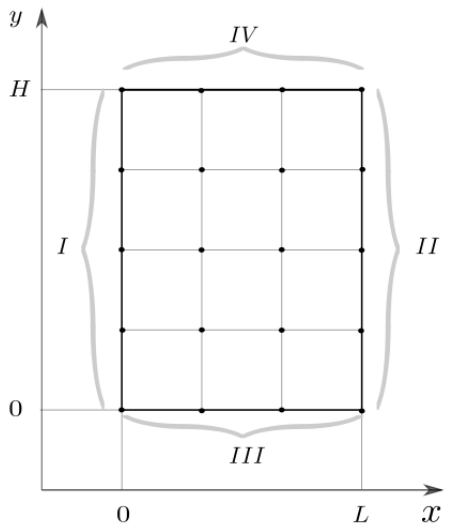


Figura 2: Aplicación de las condiciones de frontera

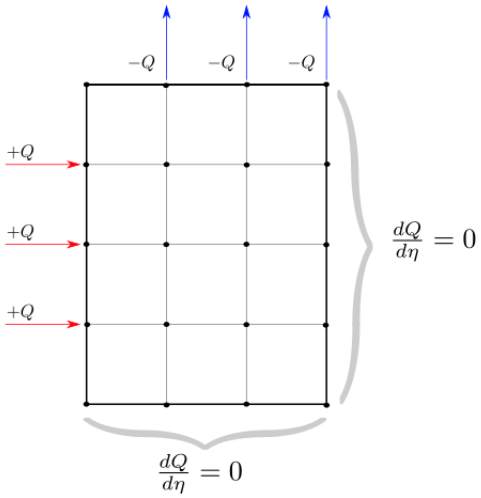


Figura 3: Entrada y salida del calor

2.3. Ejemplo

Sea el problema planteado anteriormente en una placa de un material dado. La placa será dividida en 5 filas de 4 elementos cada fila, es decir $m = 5$; $n = 4$ y $\alpha = 1 - 4r$. De nuestro sistema lineal, la matriz M quedara compuesta de la siguiente manera:

$$
\begin{bmatrix}
r & 0 & 0 & r & \alpha & r & 0 & 0 & r & & & & & & & & & & & \\
 & r & 0 & 0 & r & \alpha & r & 0 & 0 & r & & & & & & & & & & \\
 & r & 0 & 0 & r & \alpha & r & 0 & 0 & r & & & & & & & & & & \\
r & 0 & 0 & r & \alpha & r & 0 & 0 & r & & & & & & & & & & & \\
 & r & 0 & 0 & r & \alpha & r & 0 & 0 & r & & & & & & & & & & \\
 & r & 0 & 0 & r & \alpha & r & 0 & 0 & r & & & & & & & & & & \\
 & & & & r & 0 & 0 & r & \alpha & r & 0 & 0 & r & & & & & & & \\
 & & & & & r & 0 & 0 & r & \alpha & r & 0 & 0 & r & & & & & & \\
 & & & & & r & 0 & 0 & r & \alpha & r & 0 & 0 & r & & & & & & \\
 & & & & & & & & & r & 0 & 0 & r & \alpha & r & 0 & 0 & r & & \\
 & & & & & & & & & & r & 0 & 0 & r & \alpha & r & 0 & 0 & r & \\
 & & & & & & & & & & r & 0 & 0 & r & \alpha & r & 0 & 0 & r &
\end{bmatrix}
$$

Figura 4: Matriz M

En este ejemplo, los elementos con el color rojo representan la condición de borde **III**.

Aquellas filas que no tienen ningún símbolo representan una fila unicamente con ceros; para luego aplicar la condición **I**.

Las filas con color azul representan la condición **II**.

Las filas con color verde son las filas en las cuales es necesario aplicar la condición **IV** para obtener la resolución de los puntos en la frontera por donde se pierde calor. De no ser así, harían falta datos para calcular la evolución de esos puntos.

De esta manera, el sistema nos quedara de la manera:

$$u^{k+1} = Mu^k + f$$

donde f es un vector con las temperaturas de los puntos por donde ingresa calor y de esta manera aplicar la condición **I**.

2.4. Análisis del error

Sea Ψ la solución analítica, es decir sin error. Sea $\Phi = Mu^k$ la solución numérica, que contiene un error determinado.

Podemos razonar de la siguiente manera: La solución analítica es igual a la solución numérica mas el error.

$$\Psi = \Phi + \varepsilon$$

$$\Psi - \Phi = \varepsilon$$

$$\Psi - Mu^k = \varepsilon$$

Es decir, el error se hará mas chico cuando el producto Mu^k tienda a 0. Como M es una matriz, si la multiplicamos por un vector, el resultado será mas chico si los autovalores de M son menores a 1.

Al tener una matriz "grande" (en el ejemplo anterior de $20 * 20$) calcular sus autovalores de la forma tradicional, con el polinomio característico, sería un gran desafío ya que no existe método exacto para hallar las raíces de un polinomio de grado mayor a 5. Ademas, no necesitamos conocerlos exactamente; si podemos encontrar un intervalo en el cual se hayan estos valores, podremos decir si el método funciona o no. Para esto utilizaremos el teorema de Gerschgorin, que en pocas palabras nos dice entre que valores están contenidos los autovalores de una matriz.

Tenemos 2 casos no triviales para analizar.

- El primero de ellos es el caso donde el centro es $1 - 4r$ y la suma de los valores de la fila es $4r$. Si planteamos la inecuación $4r < 1$ obtenemos que $r < 1/4$, lo que nos da una margen superior para r.
- En segundo lugar tenemos el caso donde el centro es r y el resto de los valores de la fila suma $1 - 4r + 3r$. Con la inecuación $1 - 4r + 3r < 1$ obtenemos que $r > 0$. Lo que nos brinda un margen inferior para r.

Como conclusión obtenemos que $r \in (0; 1/4)$. Sin embargo esto nos deja con una interrogante: ¿qué es r? De nuestra deducción en la fórmula 4 sabemos que

$$r = \frac{\Delta t \; k}{\rho \; c \; (\Delta x)^2}$$

donde ρ es la densidad del material, c es el calor específico del material y k es la conductividad térmica del material. En otras palabras, estos tres valores son propiedades físicas del material y valores que se escapan a nuestro manejo para variar r. Por ello, para hacer que el valor de r cambie podemos cambiar Δt o Δx como lo amerite el problema. Para la resolución de este trabajo, proponemos que $r = 1/5$ y $\Delta x = 0{,}002m$. De esta manera, solo hará falta calcular Δt. Cabe aclarar que estos valores son aleatorios y subjetivos. Se podrían haber elegido otros valores sin afectar a la resolución del problema desde el punto de vista matemático.

3. Implementación

3.1. Imágenes

A continuación se pueden ver imágenes de la simulación generadas con el comando `imagesc`.

Temperatura constante En el lado izquierdo de la placa la temperatura se mantiene constante a lo largo del tiempo. Se puede ver como la placa se va calentando y a medida que pasa el tiempo la sección mas caliente pasa a ser la inferior. Esto es debido a las condiciones **II** y **III**; las cuales hacen que en esas fronteras no haya transferencia de calor con el medio y por ende aumentando su temperatura con el mas mínimo flujo de calor. Además se puede observar como la parte superior tiene un color más oscuro, denotando una menor temperatura por la perdida de calor hacia el aire.

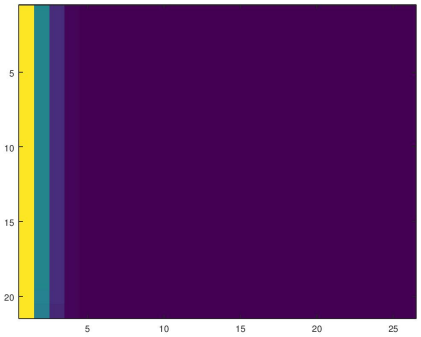


Figura 5: $T = cte$; $t = 0$

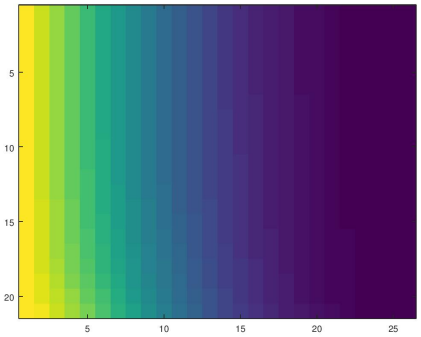


Figura 6: $T = cte$; $t = 2{,}4s$

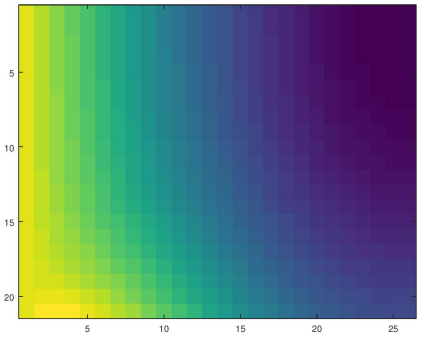


Figura 7: $T = cte$; $t = 8{,}5s$

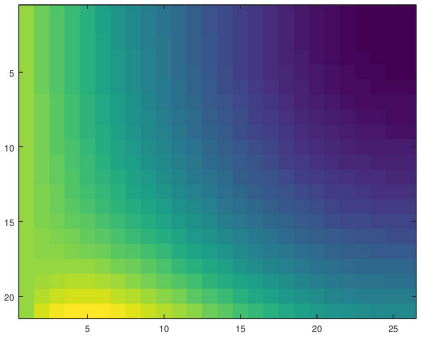


Figura 8: $T = cte$; $t = 13{,}3s$

Temperatura variable En este caso la temperatura se varia con la función:

$$f(x,t) = \begin{cases} 0 & \text{si } t < 10 \\ 50\ \sin(0{,}5t) + 70 & \text{en otro caso} \end{cases}$$

Se puede observar que cuando la función pasa a ser 0 la placa toma la misma temperatura y la empieza a perder muy despacio por el borde superior. Cabe aclarar que la forma de generar las imágenes es dinámico, es decir se le asigna un color amarillo a la parte mas caliente de esa imagen, y dos colores amarillos en distintos frames no representan la misma temperatura. Así mismo, se puede ver como en el caso anterior los puntos de mayor temperatura se acumulan en la parte inferior ya que esta aislada del medio.

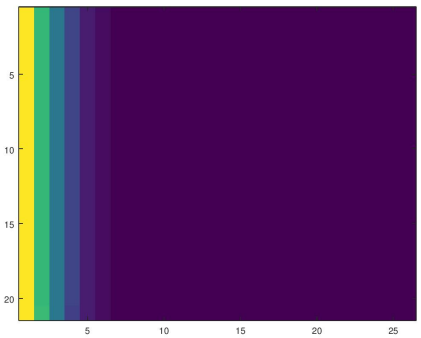


Figura 9: $T = var$; $t = 0$

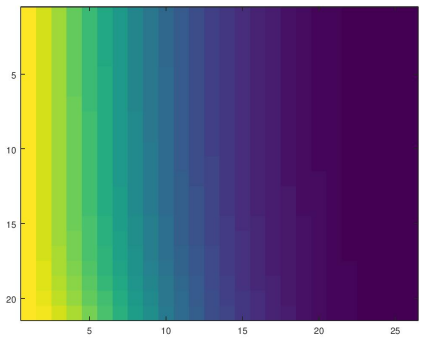


Figura 10: $T = var$; $t = 5s$

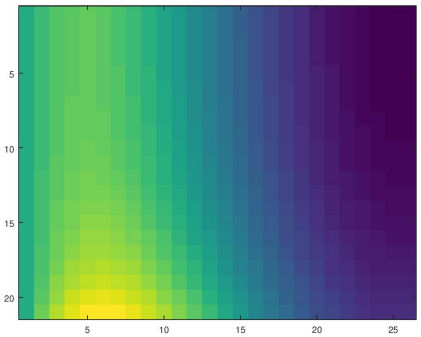


Figura 11: $T = var$; $t = 10s$

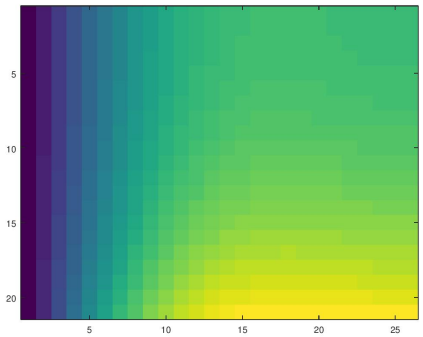


Figura 12: $T = var$; $t = 16{,}7s$

3.2. Código

Se implementó el problema en GNU Octave; teniendo en mente calcular la evolución de temperatura sobre una placa de aluminio con $40mm$ de largo $50mm$ de alto y una temperatura ambiente de $30°C$

Ir a Anexo A para ver el código o visitar `https://github.com/leo-hoet/forward_euler_2d`

4. Conclusión

Hemos desarrollado un modelo matemático que hace uso de las capacidades de los ordenadores modernos para resolver un problema que de otro modo seria imposible o muy complicado de hacerlo. Nuestro resultados, a pesar de ser correctos, contienen una cierta falla desde el principio. En particular, con la condición **II** y **III**.

Como posible desarrollo a futuro se podría plantear la modificación de estas condiciones para un modelo mas adecuado a la realidad. Además, también se podría plantear la resolución en paralelo de este problema, utilizando las capacidad de los sistemas operativos y procesadores modernos para manejar varios hilos a la vez, e incluso con posibilidad de usar GPU's especializadas en cálculos paralelos como la rama con tecnología CUDA de Nvidia para lograr disminuir considerablemente el tiempo requerido para hallar la solución.

A pesar de estas dificultades, el planteo matemático es sólido; lo que nos permite aplicarlo a otros problemas donde se encuentras planteamientos con ecuaciones diferencias sobre derivadas parciales; lo que en la ingeniería es muy común.

5. Bibliografía

5.1. Apuntes

- La ecuación del calor en 2D - O.R. Faure
- Ecuaciones en Derivadas Parciales - Versión 1.0 - UTN FRCU
- Aproximaciones en Diferencias -Versión 1.0 - UTN FRCU
- Numerical solution of partial differential equations - Dr. Louise Olsen-Kettle - The University of Queensland
- Gershgorin's Theorem for Estimating Eigenvalues - Sean Brakken-Thal

5.2. Software

- GNU Octave 4.2.2
- Texmaker 5.0.2
- Linux Mint 19.1 kernel version 4.15.0-20-generic
- Inkscape 0.92.3

Anexo

A. Cógido en octave

Para ver el código en forma online visitar: `https://github.com/leo-hoet/forward_euler_2d`

main.m : Código principal con constantes y llamadas a otras funciones.

```
##Definicion de parametros
rho = 2698.4;      ##Densidad del aluminio en kg/m^3
c = 920.887;       ##Calor especifico del aluminio en Joule/(kg*K)
k = 164;           ##Coeficiente k en watt/(metro*kelvin)
L = 0.040;         ##Largo el metros
H = 0.05;          ##Alto en metros
T_0 = 70;          ##Temperatura inicial en grados centigrados
T_a = 30;          ##Temperatura del aire en grados centigrados
h_c = 200;         ## W/(m^2 * K)
delta_x = 0.002;   ##Paso espacial en metros



delta_t = determinar_tiempo(delta_x,k,rho,c,L,H);
[M,n,m] = cargar_matriz_forward(delta_x,delta_t,k,rho,c,L,H);
U = forward_euler(M,delta_t,delta_x,H,n,h_c,k,T_a);
A = getMatrix(U,n,m);
#save_images(A);
```

determinar_tiempo.m : Determina el paso espacial para que el error sea aceptable.

```
function delta_t = determinar_tiempo (delta_x,k,rho,c,L,H)
  delta_t = (rho * c * (delta_x)^2 ) / (5 * k);
endfunction
```

cargar_matriz_forward.m : Crea la matriz del problema.

```
function [M,n,m] = cargar_matriz_forward (delta_x,delta_t,k,rho,c,L,H)

  ###Definicion de parametros
  r = (delta_t * k) / (rho * c * (delta_x)^2);

  l = [0:delta_x:L];
```

```
  h = [0:delta_x:H];


  m = length(l); #5; #
  n = length(h); #4; #

  M = zeros(n*m,n*m);  ###Creacion de la matriz
  for i = [n+2:1:n*m] ##Carga diagonal y valores adyacentes
    M(i,i) = 1-4*r;
    M(i,i-1) = r;
    if (i+1 <= n*m)
      M(i,i+1) = r;
    endif
    if (i+n < n*m)
      M(i,i+n) = r;
    endif
    M(i,i-n) = r;
  endfor

  for i = [1:n:n*m]  ##Condicion I
    M(i,:) = 0;
  endfor

  for i = [n:n:n*m] ##Condicion  II
    M(i,:) = M(i-1,:);
  endfor

  for i = [1:1:n] ## Condicion III
    M(i,:) = M(i+n,:);
  endfor

endfunction
```

forward_euler.m : Procedimiento principal de resolución del problema.

```
function U = forward_euler (M,delta_t,delta_x,H,n,h_c,k,T_a)

  ## Determinar paso temporal en segundos
  t_inicio = 1;
  t_fin = 20;
  l = [t_inicio:delta_t:t_fin]; ## Rango temporal









  ##Ampliacion de la matriz M y modificacion para condicion 4
```

```
  tamanio_original = length(M(:,1));

  M(:,tamanio_original+n-1) = 0;
  for i = [tamanio_original - n + 2 : 1 : tamanio_original]
    M(i,i+n) = M(i,i+1);
  endfor
  M(tamanio_original,:) = M(tamanio_original-1,:);


  ##Creacion de la matriz U  que contiene los resultados
  ##A medida que avanzan las columnas avanza el timepo
  U = zeros(length(M(1,:)),length(l) -1);
  for i = [tamanio_original + 2 : 1 : length(M(1,:))]
    U(i,:) = 0;#-(h_c/k) * 2 * delta_x * U(i-n,:) +  U(i-2*n
    ,:);
    M(i,i-n) = T_a * h_c/k * delta_x;
    M(i,i-2*n) = 1;
  endfor


  ##Creacion de la matriz con la temperatura incial
  F = zeros(length(M(1,:)),length(l));
  for i = [1:1:length(l)]
    for j = [1:n:length(M(:,1))]
      F(j,i) = f(delta_x * j,l(i));
    endfor
  endfor
  ###Cilco principal
  for i = [2:1:length(l)]
    U(:,i) = (M * U(:,i-1))+ F(:,i);
  endfor



endfunction
```

getMatrix.m : De la matriz de solución original U, devuelve una matriz A de $n * m$ posiciones espaciales y l posiciones temporales

```
function A = getMatrix (U,n,m)
  l = length(U(1,:))
  A = zeros(m,n,l);

  for k = [1:1:l]
    cont = 1;
    for j = [1:n:n*m]
      for i = [0:1:n-1]
        A(cont,i+1,k)= U(j+i,k);
```

```
      endfor
      cont = cont +1;
    endfor
  endfor

  endfunction
```

save_images.m : De una matriz A de entrada, crea su mapa de calor y guarda la imagen.

```
function [] = save_images (A,m)
  cont = 0001;
  for k = [1:5:length(A(1,1,:))]
    img = imagesc(A(:,:,k));
    eval(["print -djpg " mat2str(cont)]);
    cont++;
  endfor
endfunction
```

images_to_video.sh : script en bash; toma imágenes de la carpeta donde es ejecutado, las convierte en un video y elimina las imágenes.

```
#!/bin/bash
#$1 nombre de salida

#renombra las images de manera img_1.jpg --> img_0001.jpg
python3.6 rename_images.py
ffmpeg -f image2 -start_number 0001 -framerate 12  -i IMG_%04
    d.JPG -crf 15 $1
rm *.jpg
```