

Ecuaciones en derivadas parciales elípticas.

Ejemplo ecuaciones de Laplace y Poisson

Bruno Juliá-Díaz (brunojulia@ub.edu)

Dpto. Física Quàntica i Astrofísica

Facultat de Física

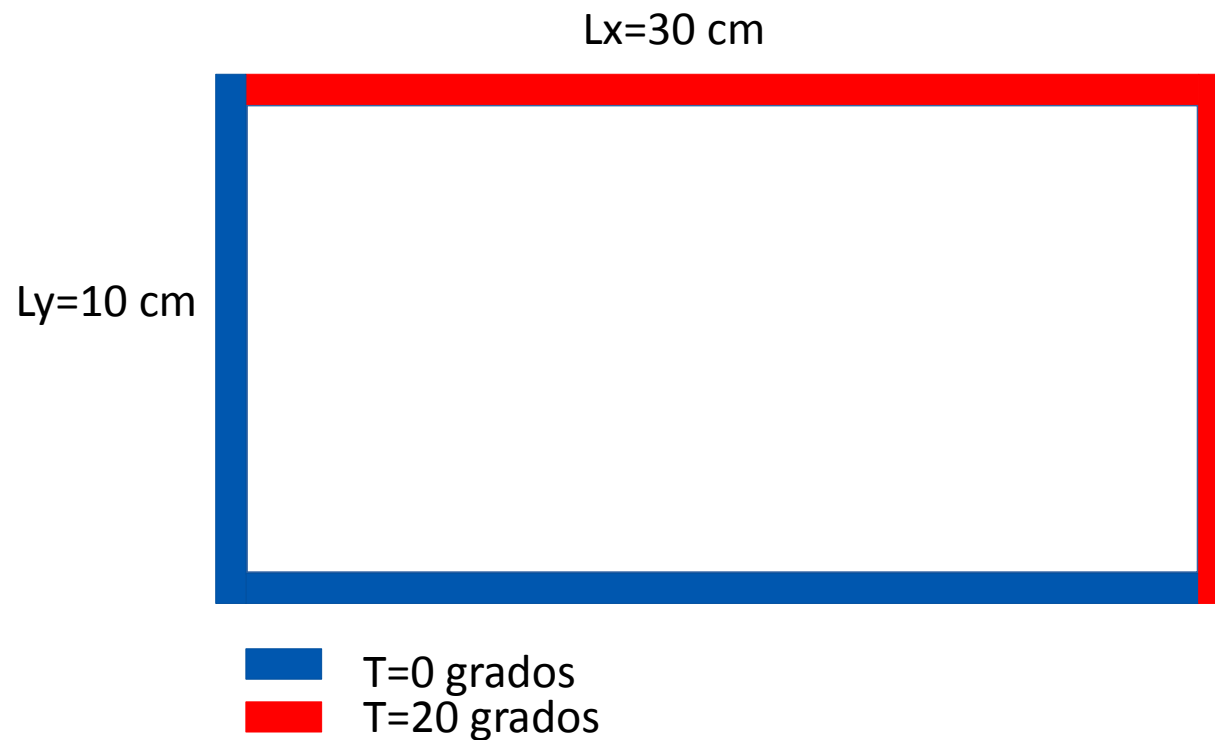
Universitat de Barcelona

Curso 2016/2017

Fuente: Gerald / Wheatley

Perfil de temperaturas

Tenemos una placa metálica bidimensional de 30cm x 10cm con una condición de contorno consistente en fijar la temperatura de la placa en sus bordes.



Perfil de temperaturas

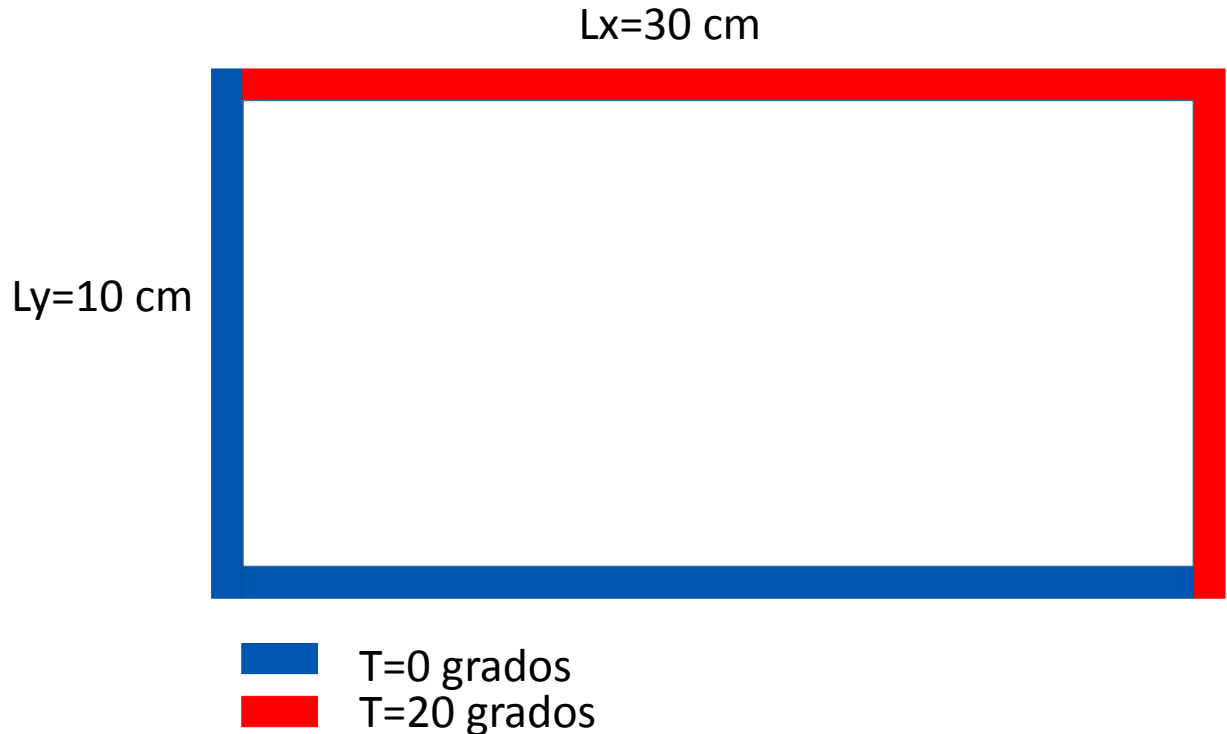
Buscamos encontrar el estado estacionario, esto es, consideramos la ecuación de difusión:

$$\frac{\partial T}{\partial t} = D \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right)$$

En el caso, $\frac{\partial T}{\partial t} = 0$

Se simplifica a resolver la ecuación de Laplace bidimensional:

$$\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) = 0$$



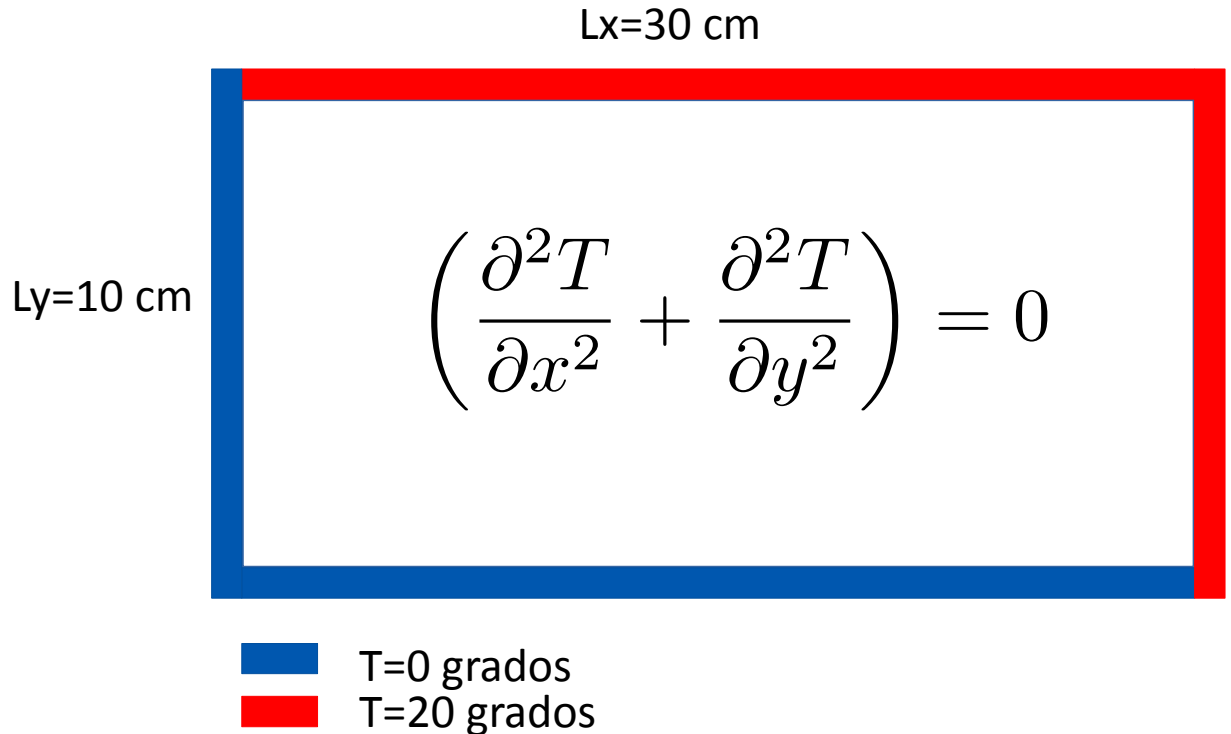
Método de paso finito

Utilizaremos un método sencillo de paso finito con el mismo valor de h para ambas coordenadas.

Tendremos:

$$x = ih \quad y = jh$$

$$\text{con } i = 0, \dots, N_x, \quad j = 0, \dots, N_y$$



Con este discretizado tendremos un conjunto de valores de la temperatura discreto, una matriz:

$$T_{i,j} \equiv T(x_i, y_j)$$

Condiciones de contorno

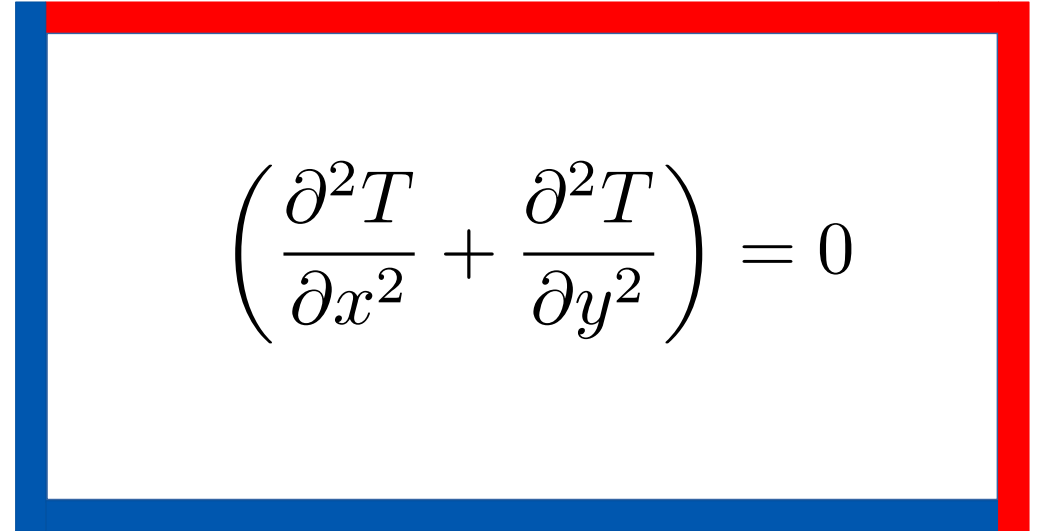
Las condiciones de contorno son en este caso:

$$T_{0,j} = T_{i,0} = 0$$

$$T_{i,N_y} = T_{N_x,j} = 20$$

$L_y = 10$ cm

$L_x = 30$ cm



■ $T = 0$ grados
■ $T = 20$ grados

El operador derivada segunda lo discretizamos en 3 puntos, con esto tenemos el conjunto de ecuaciones:

$$T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4T_{i,j} = 0$$

$$i = 1, \dots, N_x - 1 \quad j = 1, \dots, N_y - 1$$

Caso $h=5$ cm

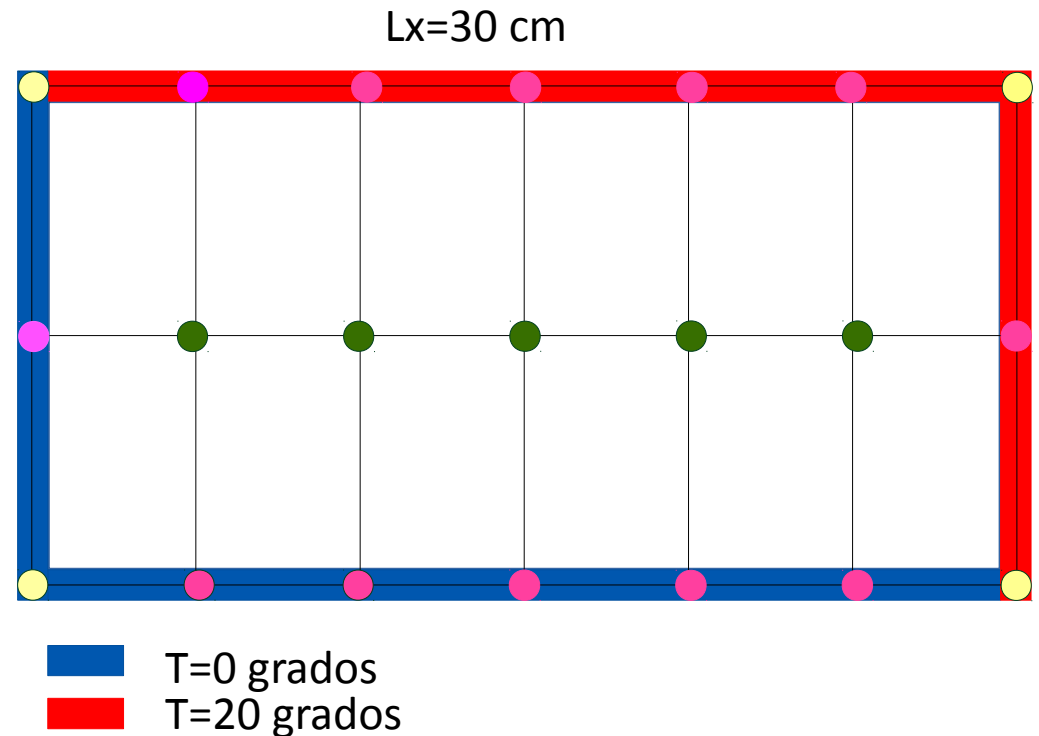
Veamos un caso muy sencillo:

$h=5$ cm

$N_x=30/5=6$

$N_y=10/5=2$

- Incognitas
- Condiciones de contorno
- Irrelevantes



Tenemos en este caso 5 ecuaciones y cinco incognitas, por ejemplo, las dos primeras son:

$$4T_{1,1} - T_{0,1} - T_{2,1} - T_{1,0} - T_{1,2} = 0$$

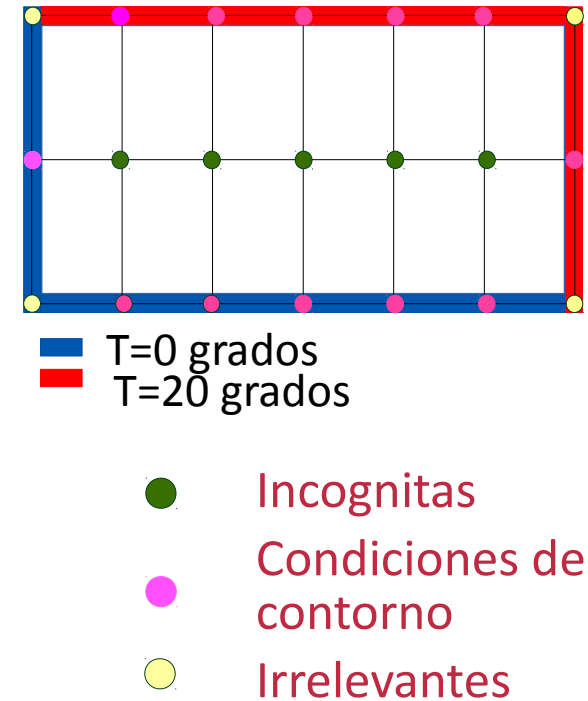
$$4T_{2,1} - T_{1,1} - T_{3,1} - T_{2,0} - T_{2,2} = 0$$

Caso $h=5$ cm

Las ecuaciones a resolver se pueden escribir como,

$$\begin{pmatrix} 4 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} T_{1,1} \\ T_{2,1} \\ T_{3,1} \\ T_{4,1} \\ T_{5,1} \end{pmatrix} = \begin{pmatrix} 20 \\ 20 \\ 20 \\ 20 \\ 40 \end{pmatrix}$$

Independientemente del valor de h utilizado, que es el que determina el tamaño de la matrix, este sistema de ecuaciones es diagonal dominante y puede solucionarse mediante métodos iterativos de forma eficiente



Caso $h=5$ cm, Gauss Seidel

Para resolverlas iterativamente es conveniente reescribirlas de una forma explícita directamente relacionada con el esquema de derivación a 2 puntos.

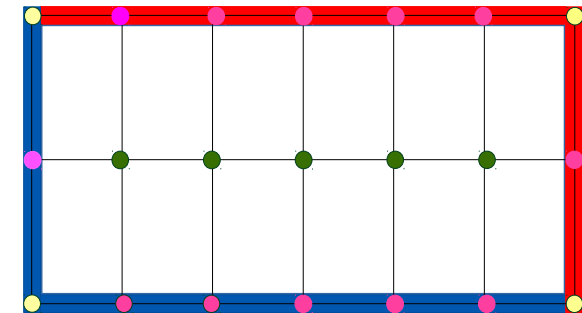
Reescribamos:

$$T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4T_{i,j} = 0$$

Como:

$$T_{i,j} = \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}}{4}$$

En esta forma podemos ahora iterar.



■ $T=0$ grados
■ $T=20$ grados

● Incógnitas
● Condiciones de contorno
● Irrelevantes

Método de Gauss Seidel

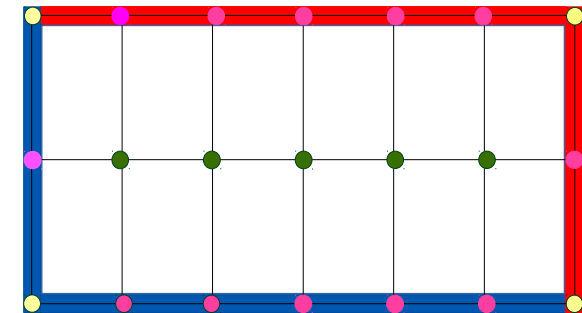
$$T_{i,j} = \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}}{4}$$

En esta forma podemos ahora iterar.

El método de Gauss-Seidel funciona del siguiente modo:

- 1) Inicializamos la matriz $T_{i,j}$ con valores, por ejemplo 1, incluyendo las condiciones de contorno.
- 2) Utilizamos la ecuación de arriba para recalcular todos los valores:

$$T_{i,j}^{\text{nuevo}} = \frac{T_{i+1,j}^{\text{old}} + T_{i-1,j}^{\text{old}} + T_{i,j+1}^{\text{old}} + T_{i,j-1}^{\text{old}}}{4}$$

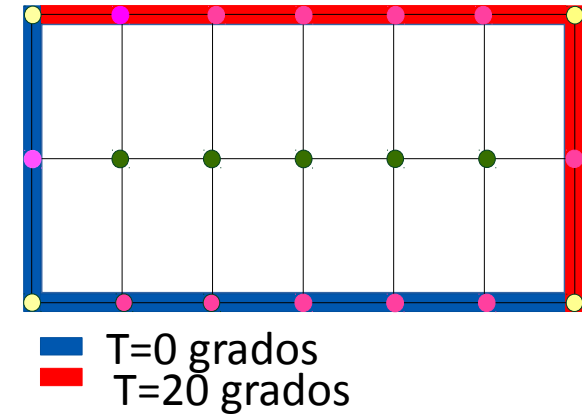


■ T=0 grados
■ T=20 grados

● Incógnitas
● Condiciones de contorno
● Irrelevantes

Caso $h=5$ cm, Gauss Seidel, convergencia

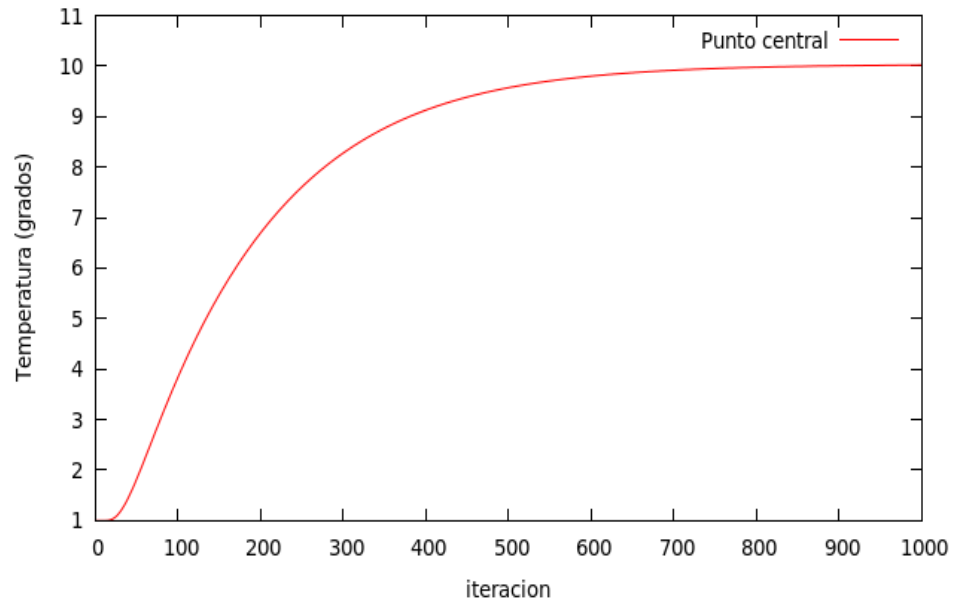
| lte | T11 | T21 | T31 | T41 | T51 |
|-----|---------|---------|---------|----------|----------|
| 0 | 1.00000 | 1.00000 | 1.00000 | 1.00000 | 1.00000 |
| 1 | 5.25000 | 5.50000 | 5.50000 | 5.50000 | 10.25000 |
| 2 | 6.37500 | 7.68750 | 7.75000 | 8.93750 | 11.37500 |
| 3 | 6.92188 | 8.53125 | 9.15625 | 9.78125 | 12.23438 |
| 4 | 7.13281 | 9.01953 | 9.57812 | 10.34766 | 12.44531 |
| 5 | 7.25488 | 9.17773 | 9.84180 | 10.50586 | 12.58691 |
| 6 | 7.29443 | 9.27417 | 9.92090 | 10.60718 | 12.62646 |
| 7 | 7.31854 | 9.30383 | 9.97034 | 10.63684 | 12.65179 |
| 8 | 7.32596 | 9.32222 | 9.98517 | 10.65553 | 12.65921 |
| 9 | 7.33055 | 9.32778 | 9.99444 | 10.66109 | 12.66388 |
| 10 | 7.33195 | 9.33125 | 9.99722 | 10.66458 | 12.66527 |
| 11 | 7.33281 | 9.33229 | 9.99896 | 10.66562 | 12.66615 |
| 12 | 7.33307 | 9.33294 | 9.99948 | 10.66628 | 12.66641 |



Hemos resuelto correctamente las ecuaciones.

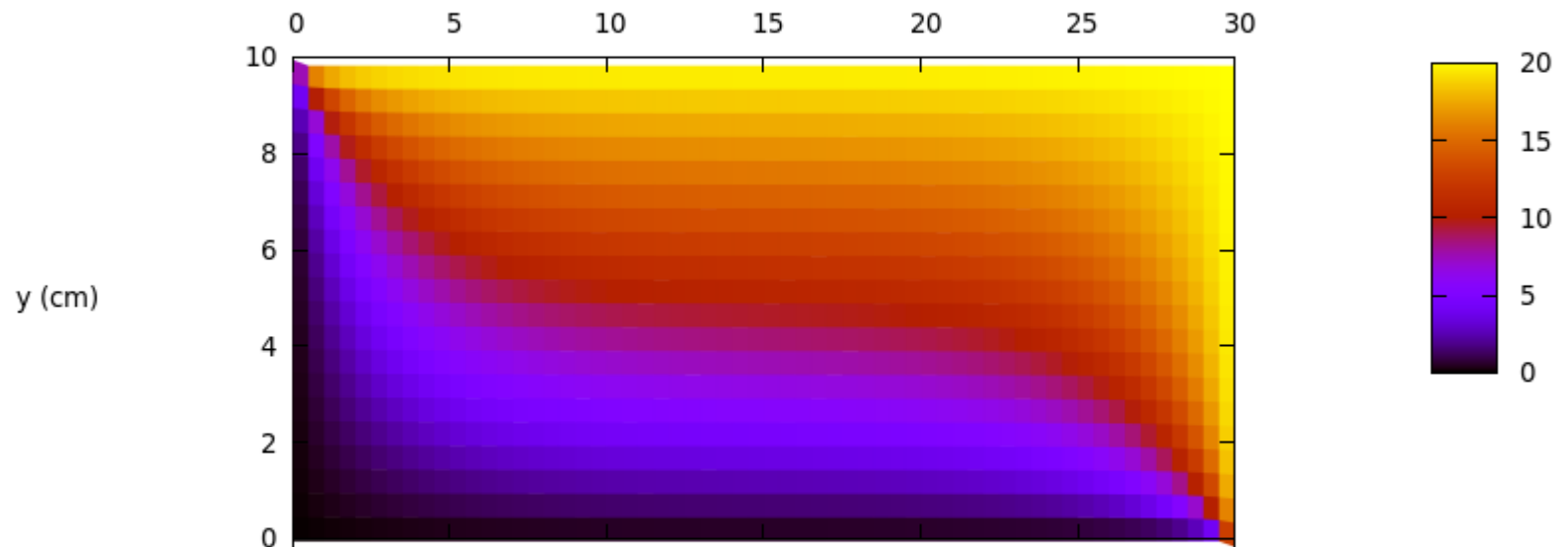
¿Hemos resuelto correctamente el problema original?

Caso $h=0.5$ cm, Gauss Seidel, convergencia

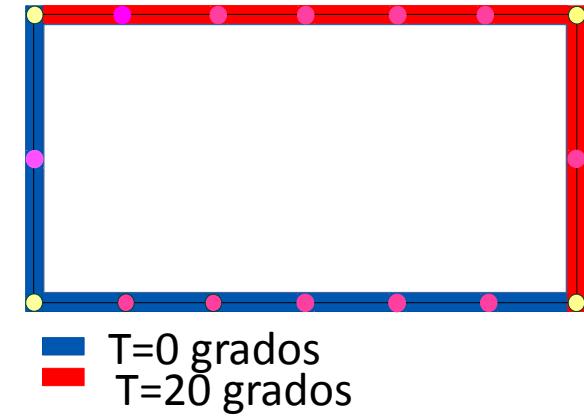
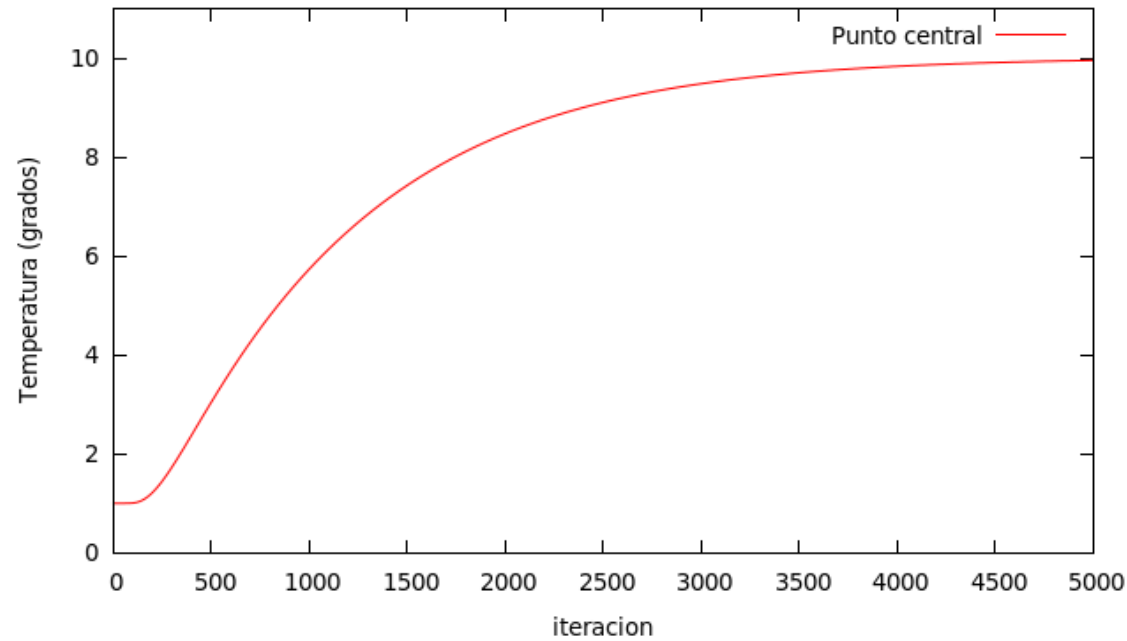


Temperatura

x (cm)

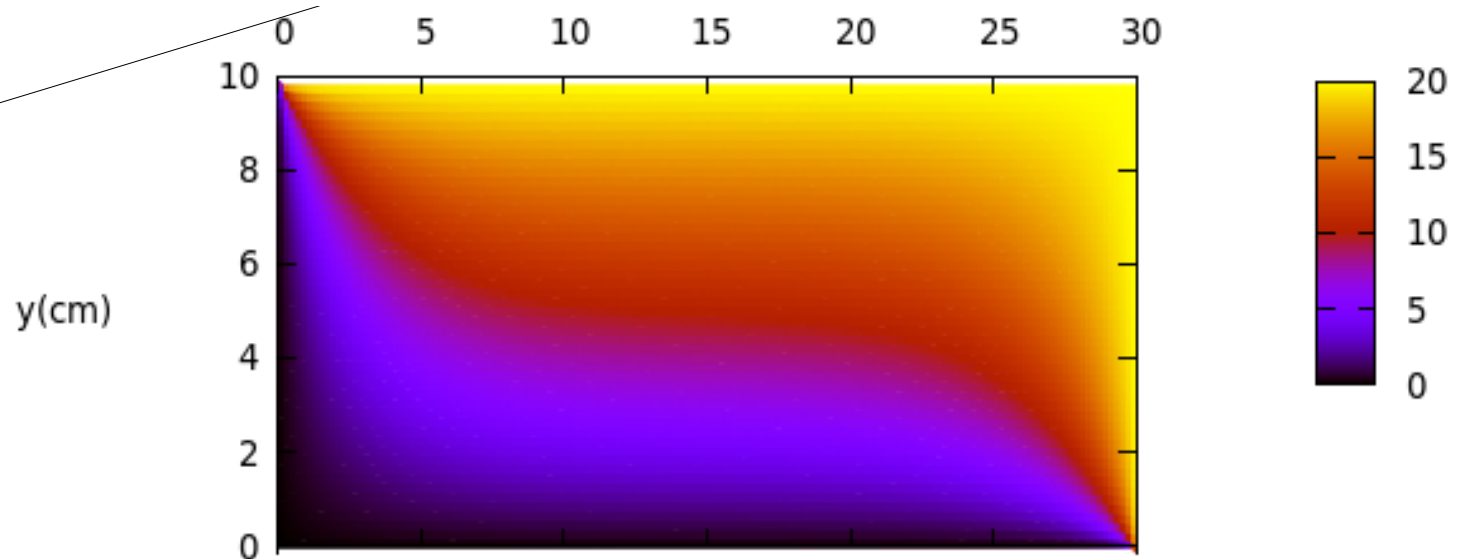


Caso $h=0.2$ cm, Gauss Seidel, convergencia

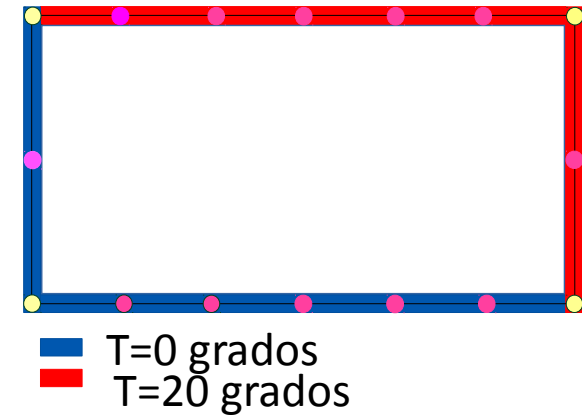
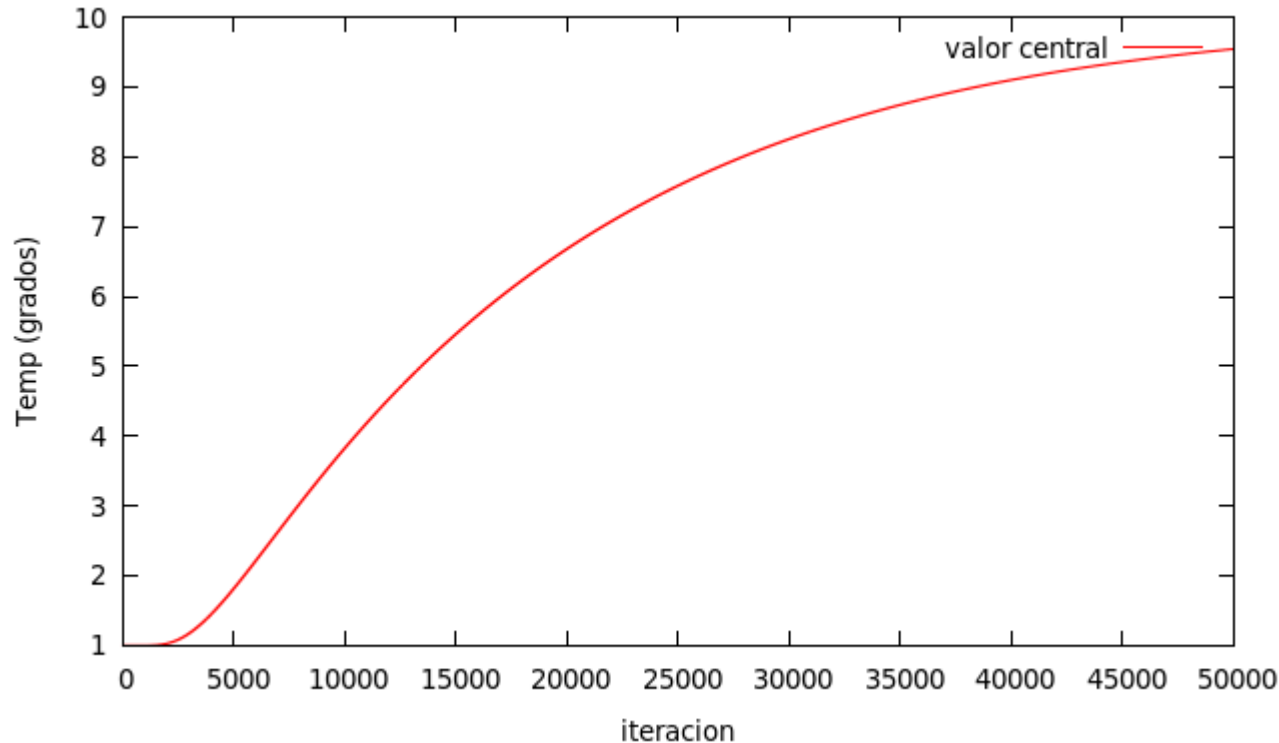


Muchas más iteraciones
necesarias para obtener
un error parecido en la
solución.

Número de ecuaciones:
 $149 \times 49 = 7301$



Caso $h=0.05$ cm, Gauss Seidel, convergencia



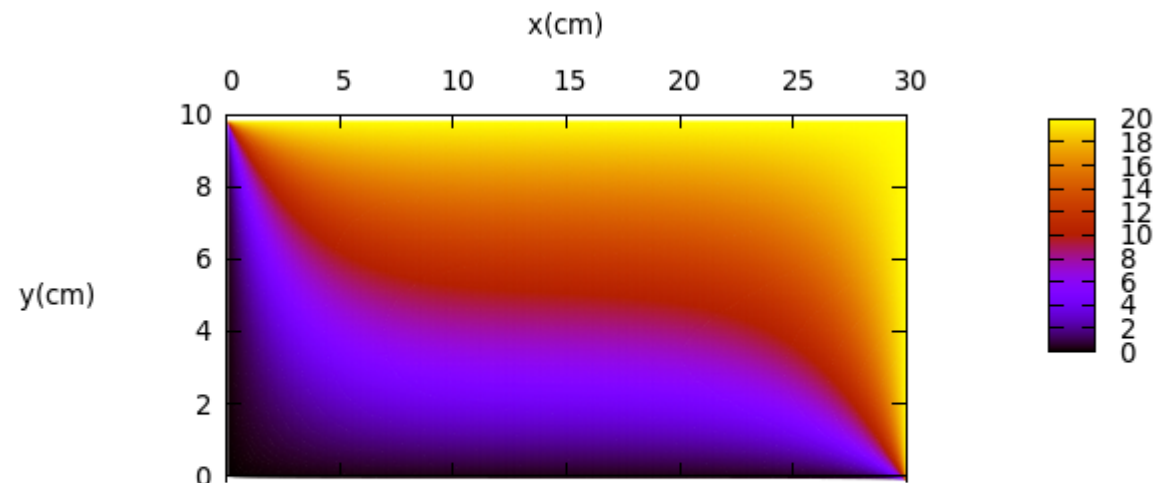
Temperatura

Tamaño:
300 x 200

Aún no está convergido

Número de ecuaciones:
599 x 199 = 119201

Tiempo de cálculo:
6 minutos
Gfortran -O3



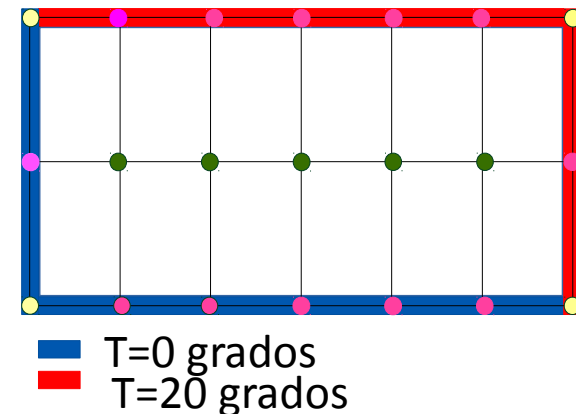
Método de Jacobi

$$T_{i,j} = \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}}{4}$$

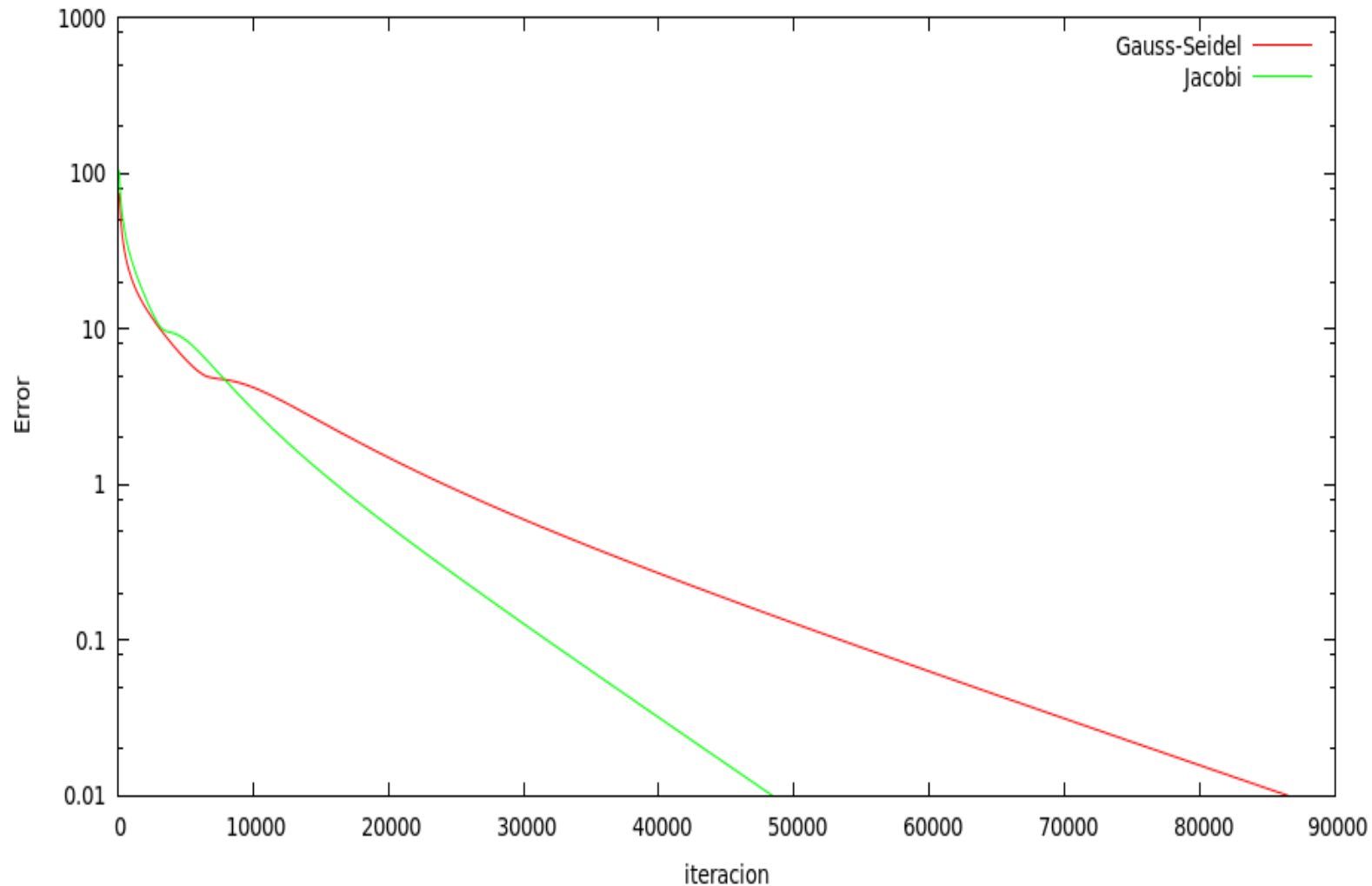
El método de **Jacobi** funciona del siguiente modo:

- 1) Inicializamos la matriz $T_{i,j}$ con valores, por ejemplo 1, incluyendo las condiciones de contorno.
- 2) Utilizamos la ecuación de arriba para recalcular todos los valores. Pero ahora, utilizamos como valores “old” los valores ya recalculados en cada iteración:

$$T_{i,j}^{\text{nuevo}} = \frac{T_{i+1,j}^{\text{nuevo}} + T_{i-1,j}^{\text{nuevo}} + T_{i,j+1}^{\text{nuevo}} + T_{i,j-1}^{\text{nuevo}}}{4}$$



Caso $h=0.05$ cm, Jacobi vs Gauss Seidel

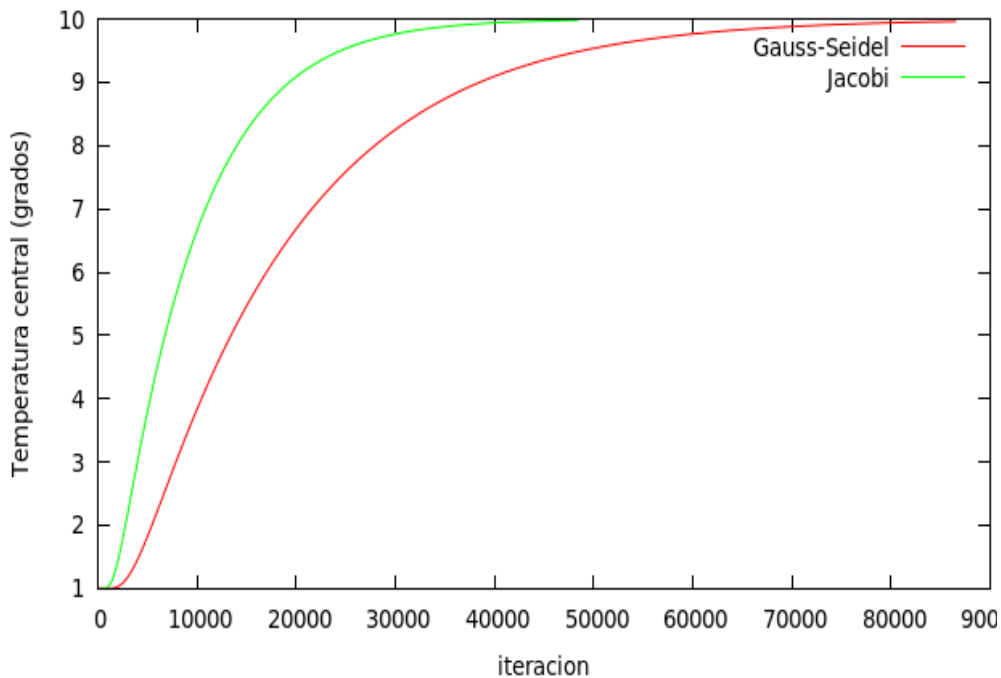


Tamaño:
300 x 200

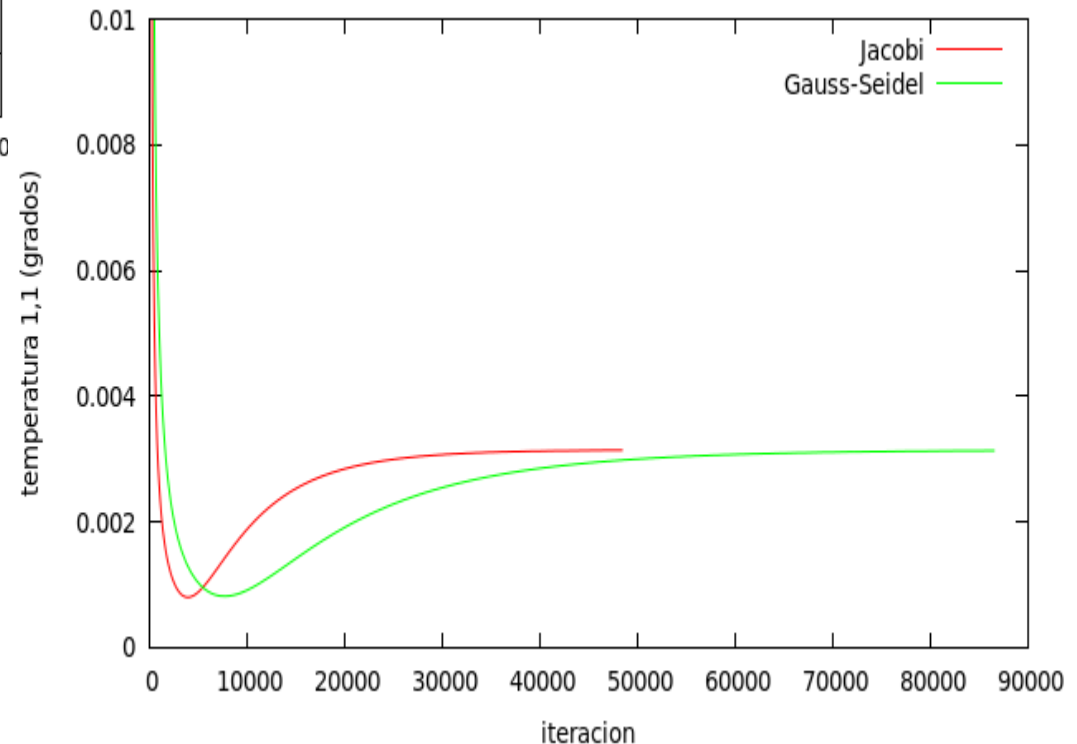
Número de ecuaciones
 $599 \times 199 = 119201$

Tiempo de cálculo:
1m desktop
Gfortran -O3

Caso $h=0.05$ cm, Jacobi vs Gauss Seidel



Temperatura central

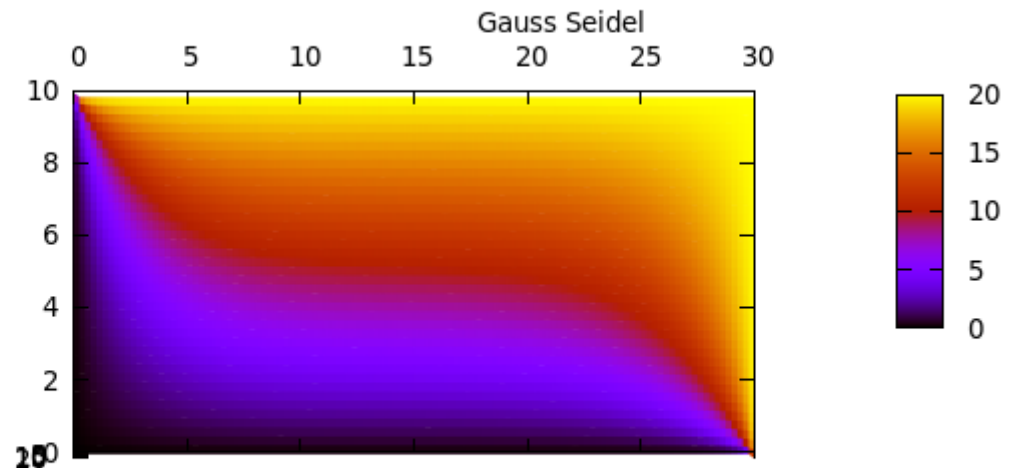
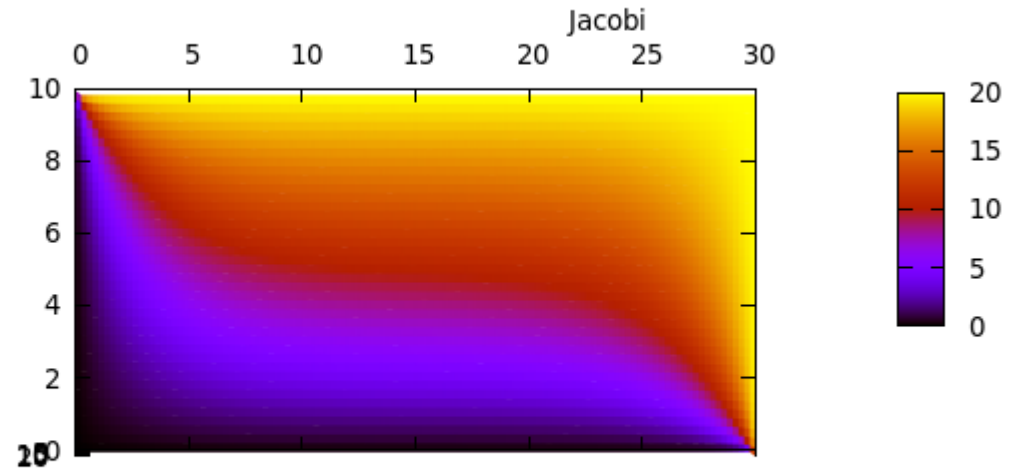


Tamaño:
300 x 200

Número de ecuaciones:
 $599 \times 199 = 119201$

Tiempo de cálculo:
6 minutos
Gfortran -O3

Caso $h=0.05$ cm, Jacobi vs Gauss Seidel



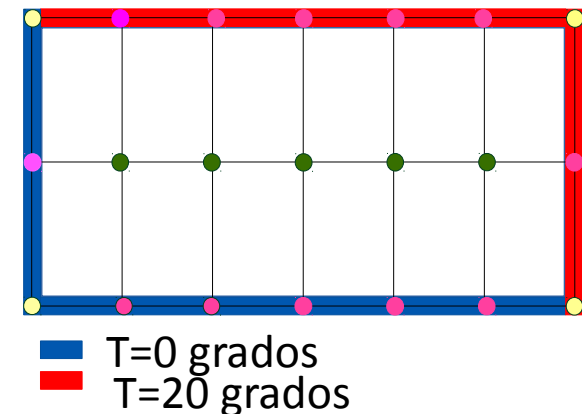
Tamaño:
300 x 200

Número de ecuaciones:
599 x 199 = 119201

Tiempo de cálculo:
6 minutos
Gfortran -O3

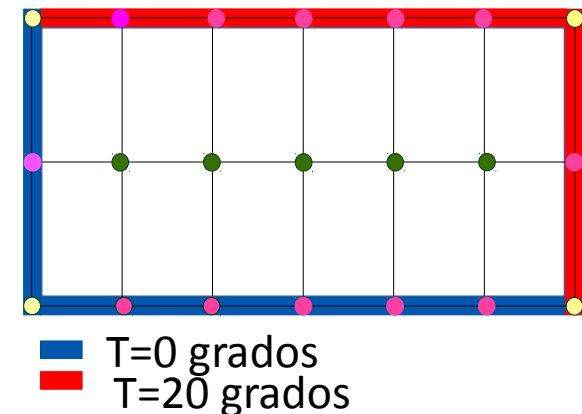
Algoritmo Gauss Seidel

- > Inicializamos dos matrices de temperaturas, $T_{NEW}(N_x, N_y)$ y $T_{OLD}(N_x, N_y)$, Incluyendo condiciones de contorno
- > Arrancamos un bucle hasta obtener una determinada tolerancia
 - > Recorremos todo el interior de la matriz de temperaturas, e.g. con dos bucles anidados,
 - + Calculamos T_{NEW} a partir de T_{OLD}
 - + Calculamos el error cometido, estimándolo a partir de las diferencias entre T_{NEW} y T_{OLD}
 - > Asignamos $T_{OLD} = T_{NEW}$ y seguimos hasta tener la tolerancia



Algoritmo a la Jacobi

- > Inicializamos una matriz de temperaturas, $T_{NEW}(N_x, N_y)$. Incluyendo condiciones de contorno
- > Arrancamos un bucle hasta obtener una determinada tolerancia
 - > Recorremos todo el interior de la matriz de temperaturas, e.g. con dos bucles anidados,
 - + Calculamos $T_{NEW}(i,j)$ a partir de T_{NEW}
 - + Calculamos el error cometido, estimandolo a partir de las diferencias entre $T_{NEW}(i,j)$ y $T_{NEW}(i,j)$ previo
 - > Seguimos hasta tener la tolerancia



En fortran (inicializamos)

```
implicit none
double precision dx, lx, ly
integer nx, ny, nkmax, igauss

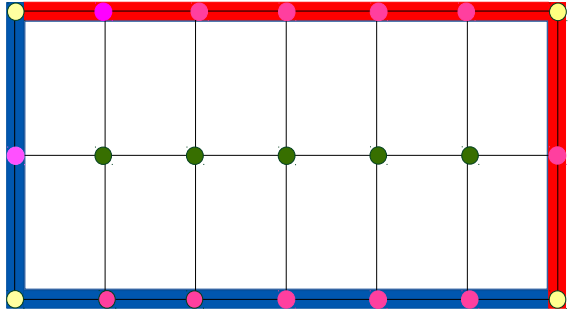
c paso en x e y
parameter (dx=0.05d0)

c longitud de la caja en cm
parameter (lx=30.d0)
parameter (ly=10.d0)

c numero de puntos en cada dirección
parameter (nx=int(lx/dx))
parameter (ny=int(ly/dx))

c numero de iteraciones de gauss seidel
parameter (nkmax=100000)

c matrices de temperaturas
double precision tnew(0:nx,0:ny), told(0:nx,0:ny), error, tol
integer i, j, k
```



■ T=0 grados
■ T=20 grados

En fortran, condiciones de contorno

```
Igauss=1
```

```
c boundaries
```

```
do i=0,nx
```

```
  told(i,0)=0.d0
```

```
  told(i,ny)=20.d0
```

```
enddo
```

```
do j=0,ny
```

```
  told(0,j)=0.d0
```

```
  told(nx,j)=20.d0
```

```
Enddo
```

```
c Arrancamos todas las temperaturas a 1 (Excepto el contorno)
```

```
do i=1,nx-1
```

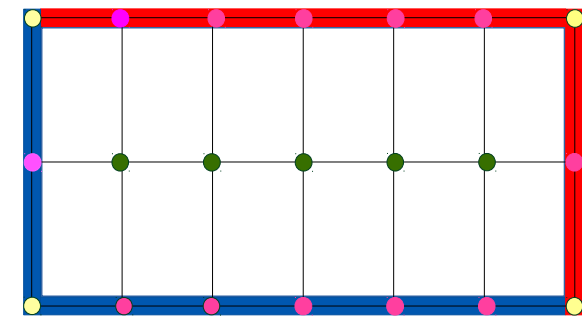
```
do j=1,ny-1
```

```
c todos menos los bordes puestos a 1
```

```
  told(i,j)=1.d0
```

```
enddo
```

```
enddo
```



■ T=0 grados
■ T=20 grados

En fortran, Gauss-Seidel / Jacobi

```
c iteracion de Gauss Seidel
```

```
    k=0
```

```
10    continue
```

```
    k=k+1
```

```
c
```

```
    if (mod(k,100).eq.0) write(33,333) k-1,tnew(nx/2,ny/2),tnew(2,2)
```

```
333    format(I5,2x,(20(e12.5,2x)))
```

```
c calcula los puntos siguientes
```

```
    do i=1,nx-1
```

```
        do j=1,ny-1
```

```
            If (igauss.eq.1) then
```

```
c Gauss Seidel
```

```
        tnew(i,j)=(told(i+1,j)+told(i-1,j)+told(i,j+1)+told(i,j-1))/4.d0
```

```
        else
```

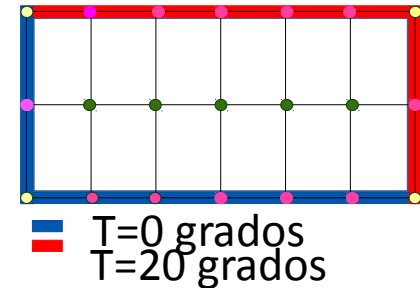
```
c Jacobi
```

```
        tnew(i,j)=(tnew(i+1,j)+tnew(i-1,j)+tnew(i,j+1)+tnew(i,j-1))/4.d0
```

```
    endif
```

```
    enddo
```

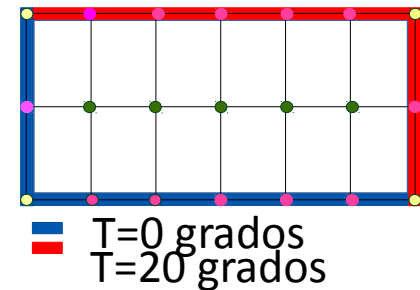
```
enddo
```



En fortran, criterio de convergencia

```
c calcula error para ver la convergencia y
c told=tnew para la proxima iteracion
    error=0.d0
    do i=1,nx-1
    do j=1,ny-1
        error=error+abs(told(i,j)-tnew(i,j))/(told(i,j)+tnew(i,j))
        told(i,j)=tnew(i,j)
    enddo
    enddo
    if (mod(k,100).eq.0) write(50,*) k,error
c check de convergencia
    if (error.lt.tol.or.k.eq.nkmax) goto 20
    goto 10

c Iteracion de Gauss Seidel terminada
20      continue
```



Método de sobre-relajación sucesiva

$$T_{i,j} = \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}}{4}$$

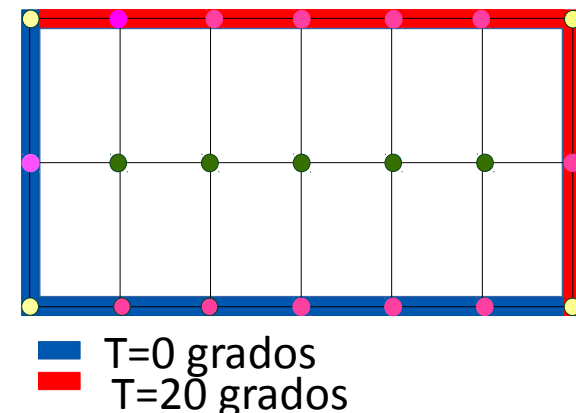
Existen algunos métodos que también funcionan en este problema que son los métodos de sobre-relajación de Southwell.

Una forma de derivarlos es la siguiente. Comencemos con esta versión:

$$T_{i,j}^{\text{nuevo}} = \frac{T_{i+1,j}^{\text{old}} + T_{i-1,j}^{\text{nuevo}} + T_{i,j+1}^{\text{old}} + T_{i,j-1}^{\text{nuevo}}}{4}$$

$$T_{i,j}^{\text{nuevo}} = T_{i,j}^{\text{old}} + \frac{T_{i+1,j}^{\text{old}} + T_{i-1,j}^{\text{nuevo}} + T_{i,j+1}^{\text{old}} + T_{i,j-1}^{\text{nuevo}} - 4T_{i,j}^{\text{old}}}{4}$$

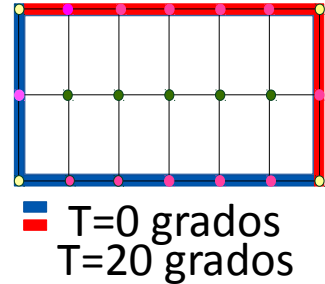
Corrección



Quando haya convergido, la corrección sera **ZERO**

Método de sobre-relajación sucesiva

$$T_{i,j}^{\text{nuevo}} = T_{i,j}^{\text{old}} + \frac{T_{i+1,j}^{\text{old}} + T_{i-1,j}^{\text{nuevo}} + T_{i,j+1}^{\text{old}} + T_{i,j-1}^{\text{nuevo}} - 4T_{i,j}^{\text{old}}}{4}$$



El método consiste en aplicar un factor multiplicativo a la corrección, para hacer que converja más rápidamente:

$$T_{i,j}^{\text{nuevo}} = T_{i,j}^{\text{old}} + \omega \frac{T_{i+1,j}^{\text{old}} + T_{i-1,j}^{\text{nuevo}} + T_{i,j+1}^{\text{old}} + T_{i,j-1}^{\text{nuevo}} - 4T_{i,j}^{\text{old}}}{4}$$

El parámetro ω tiene su valor óptimo entre 1 y 2. Para cada problema hay un valor óptimo. Algunos métodos refinan el valor de ω dependiendo de los valores de las primeras iteraciones.

Efecto de ω

Estudiamos la convergencia del valor central como función de ω

$H=0.05$ cm

Criterio de convergencia: máximo error local 0.000001d0

ω Iteración (aprox)

1.0 55000

1.2 40000

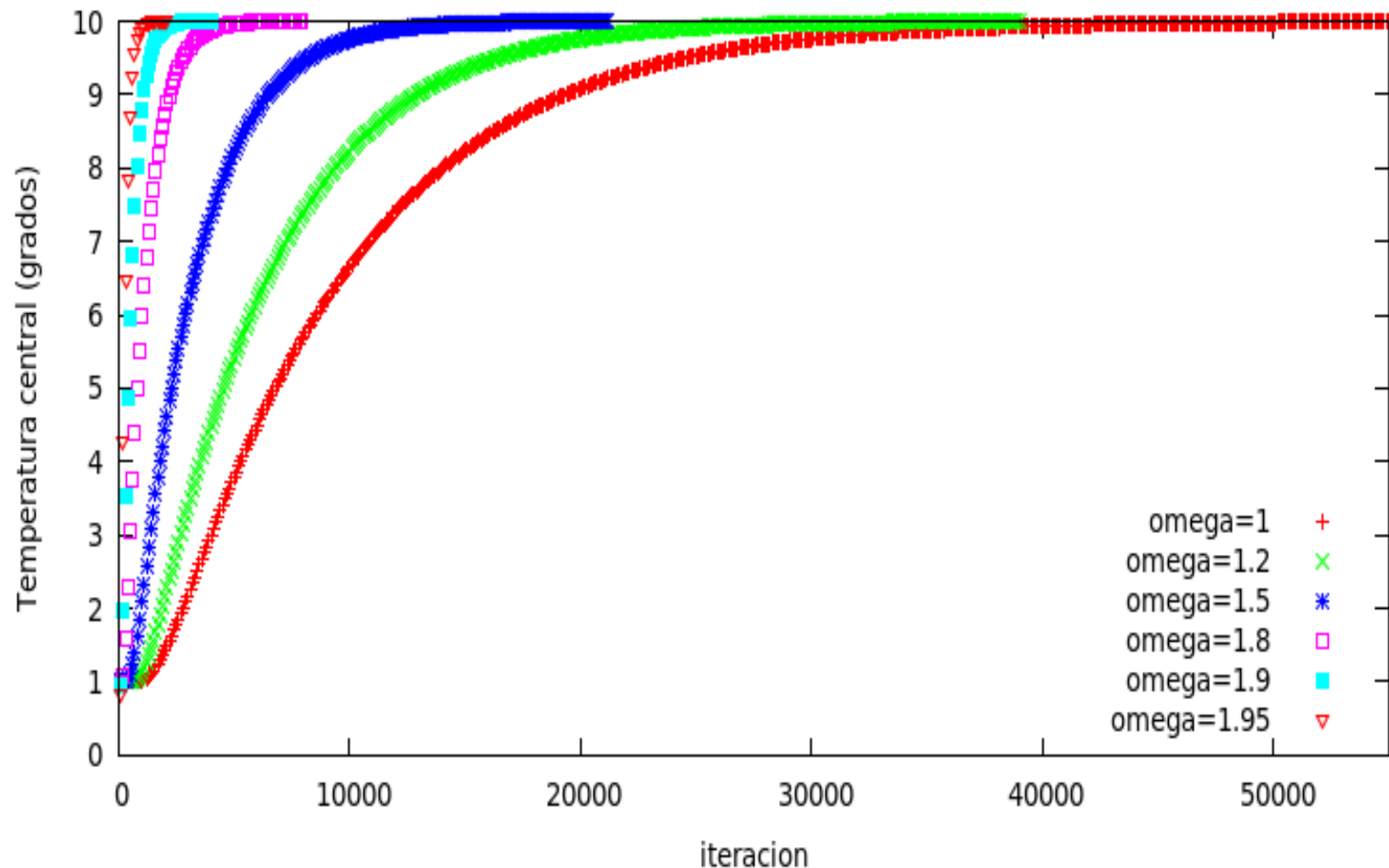
1.5 21000

1.8 8000

1.9 4300

1.95 2500

2.0 INESTABLE



Fórmulas de orden superior

La fórmula a 5 puntos se suele escribir con la siguiente notación: (error h^2)

$$\nabla^2 T_{i,j} = \frac{1}{h^2} \begin{Bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{Bmatrix} T_{i,j} = 0$$

$$T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4T_{i,j} = 0$$

Se puede derivar una fórmula a 9 puntos, con estructura: (error h^6)

$$\nabla^2 T_{i,j} = \frac{1}{6h^2} \begin{Bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{Bmatrix} T_{i,j} = 0$$

$$T_{i+1,j+1} + T_{i-1,j+1} + T_{i+1,j-1} + T_{i-1,j-1} + 4T_{i+1,j} + 4T_{i-1,j} + 4T_{i,j+1} + 4T_{i,j-1} - 20T_{i,j} = 0$$

Ecuación de Poisson

Las mismas técnicas pueden aplicarse para resolver la ecuación de Poisson 2D

$$\nabla^2 u(x, y) = \rho(x, y)$$

Con la regla de 5 puntos tendremos:

$$\nabla^2 u_{i,j} = \frac{1}{h^2} \begin{Bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{Bmatrix} u_{i,j} = \rho_{i,j}$$

e.g
Fuente de calor
Cargas
Masas
....

Y podremos utilizar los mismos métodos iterativos para resolver el sistema de ecuaciones.

Poisson

Veamos un ejemplo, siguiendo el Gerald/Wheatley, de ecuación de Poisson

Resolvamos el problema: ($\phi(x,y)=0$ en el contorno)

$$\nabla^2 \phi(x, y) + 2 = 0$$

Aparece la “h”

Utilizando la fórmula a 5 puntos habitual tendremos:

$$\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} + 2h^2 - 4\phi_{i,j} = 0$$

Que hacemos explícita para utilizar métodos iterativos:

$$\phi_{i,j} = \frac{\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} + 2h^2}{4}$$

Poisson

Utilicemos Gauss-Seidel, en este caso:

$$\phi_{i,j}^{k+1} = \frac{\phi_{i+1,j}^k + \phi_{i-1,j}^k + \phi_{i,j+1}^k + \phi_{i,j-1}^k + 2h^2}{4}$$

Indica la iteración

Indica la posición en la malla 2D

Si quisiéramos utilizar Jacobi tendríamos:

(si suponemos que recorremos la matriz con i y j ascendentes)

$$\phi_{i,j}^{k+1} = \frac{\phi_{i+1,j}^k + \phi_{i-1,j}^{k+1} + \phi_{i,j+1}^k + \phi_{i,j-1}^{k+1} + 2h^2}{4}$$

Poisson, G-S

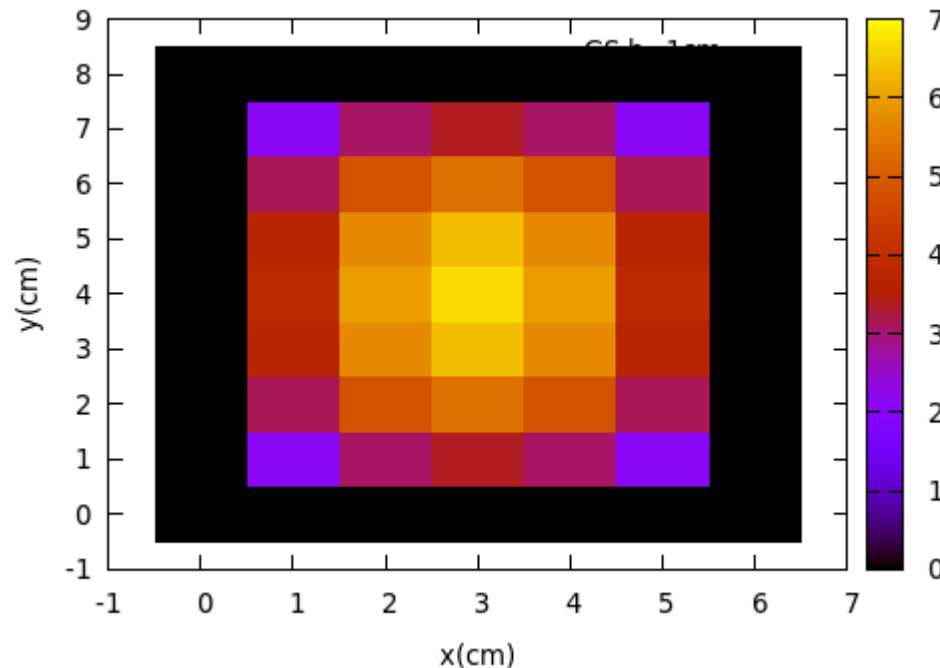
Utilicemos Gauss-Seidel, en este caso:

$$\phi_{i,j}^{k+1} = \frac{\phi_{i+1,j}^k + \phi_{i-1,j}^k + \phi_{i,j+1}^k + \phi_{i,j-1}^k + 2h^2}{4}$$

h=1 cm,

GS requiere (sin tener en cuenta simetrías), unas **80** iteraciones para tolerancia 0.001

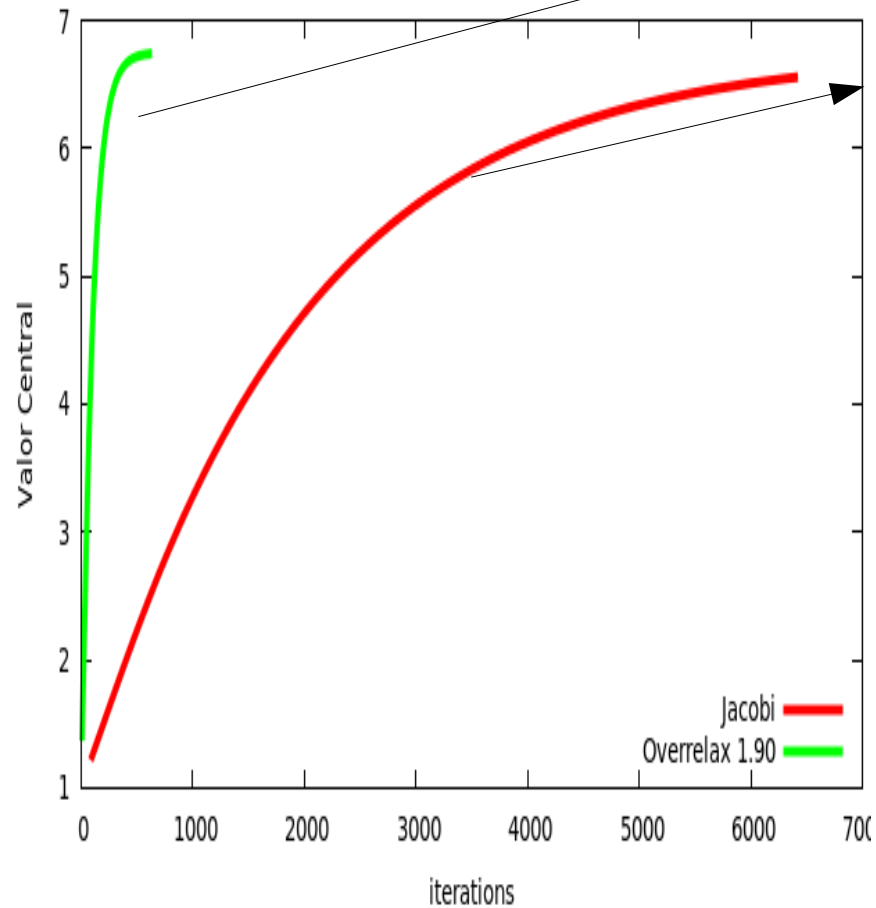
Jacobi requiere unas **25** para tener la misma precisión.



Valor de $\Phi(x,y)$ (en color)

Poisson, G-S

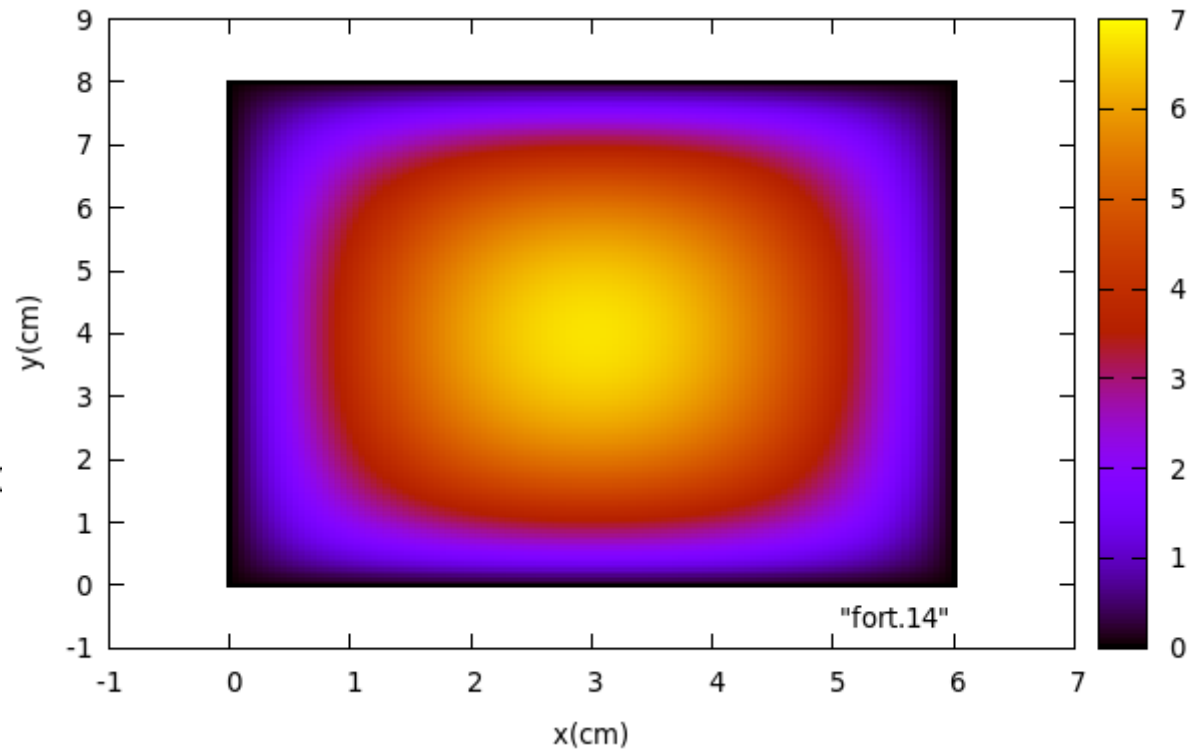
$H=0.05$ cm



Sobre Relajación $\omega=1.9$

Jacobi

Valor de $\Phi(x,y)$ (en color)

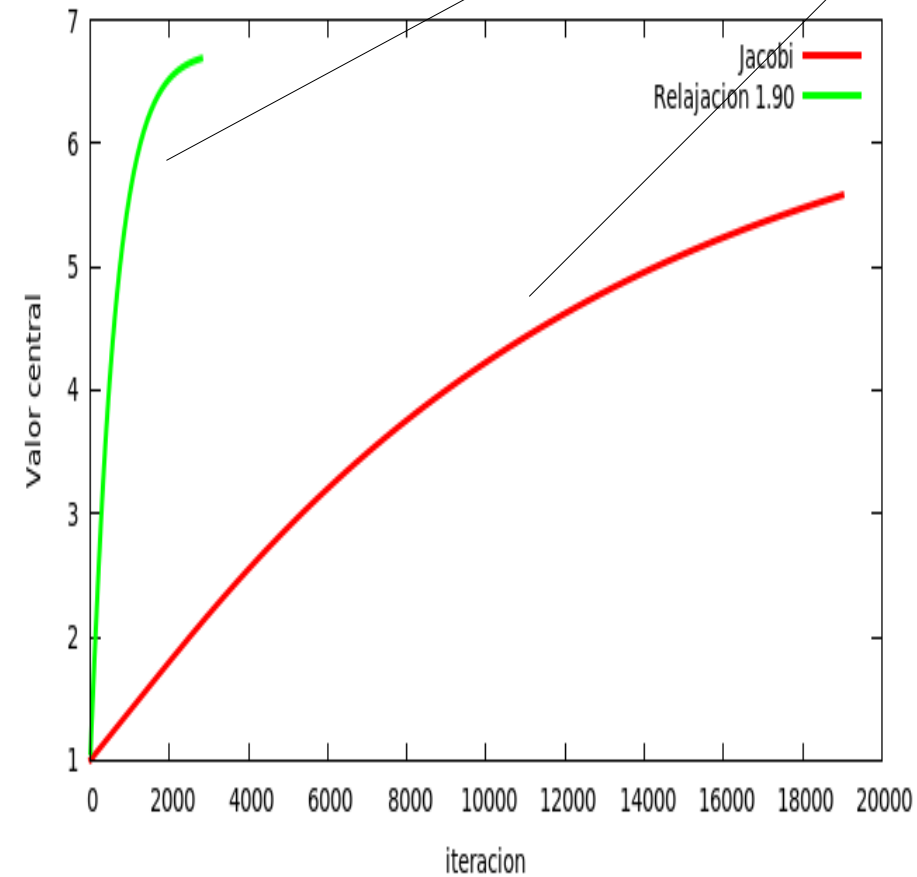


Poisson, G-S

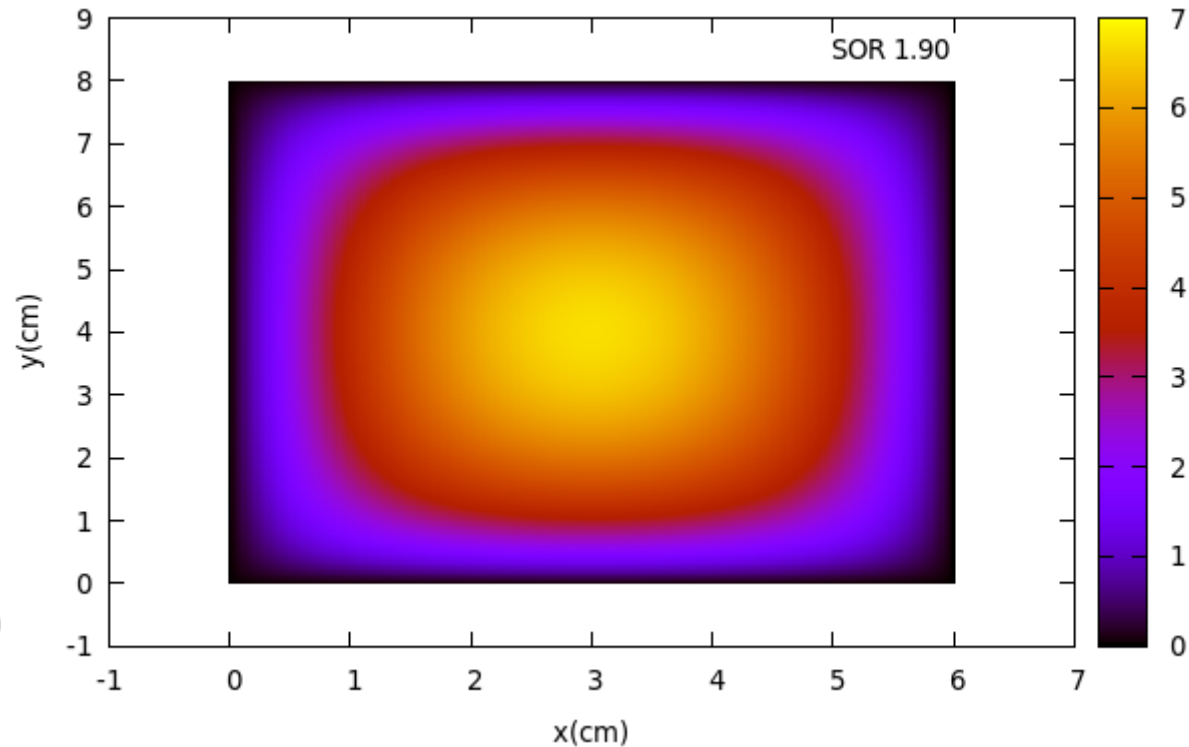
$H=0.02$ cm

Sobre Relajacion omega=1.9

Jacobi



Valor de $\Phi(x,y)$ (en color)



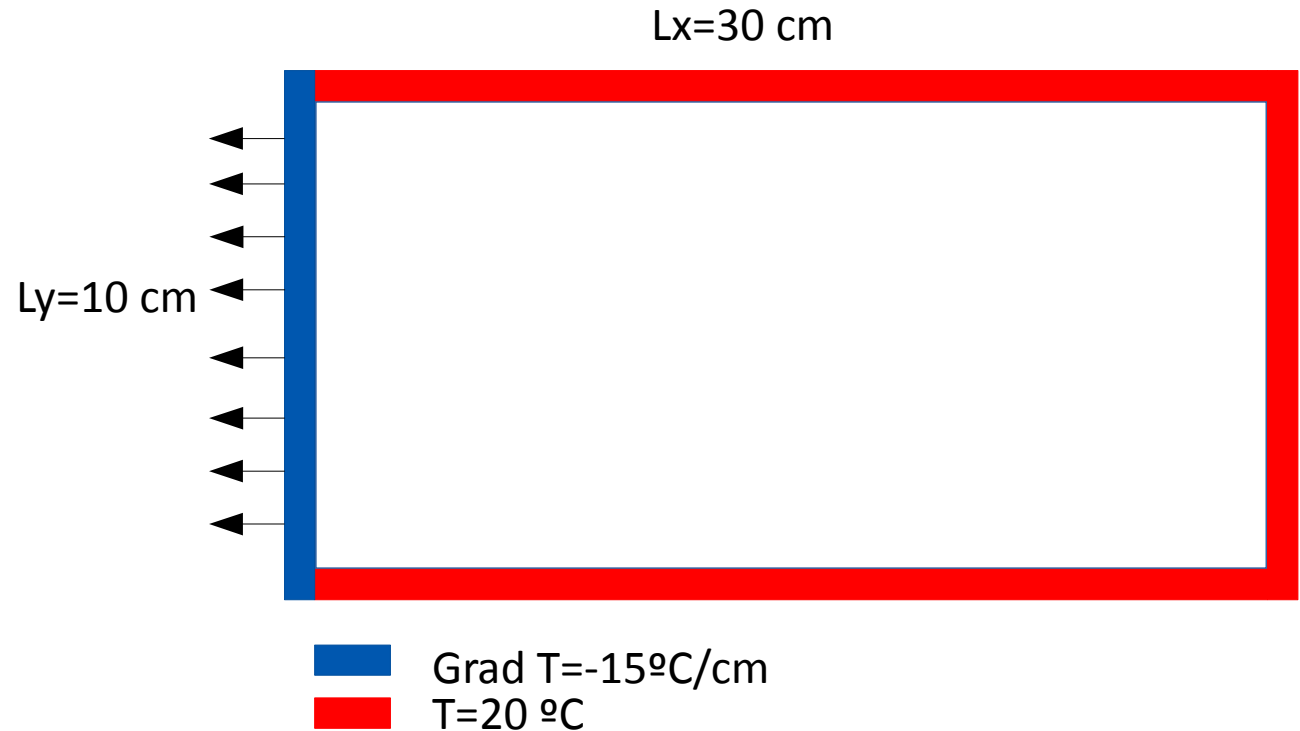
Condiciones de Neumann

Hasta ahora hemos considerado condiciones de contorno de Dirichlet, ¿Cómo podemos incorporar condiciones de Neumann?

Imaginemos el problema con Condiciones:

$$\left. \frac{\partial T(x, y)}{\partial x} \right|_{x=0} = -15^\circ\text{C}/\text{cm}$$

$$T(x, y = 0) = T(x = L_x, y) = T(x, y = L_y) = 20^\circ\text{C}$$



Condiciones de Neumann

Para implementar la condición de Neumann:

- > Añadimos una columna de puntos (en este caso)
- > Utilizamos una formula a dos puntos para la derivada en esos puntos:

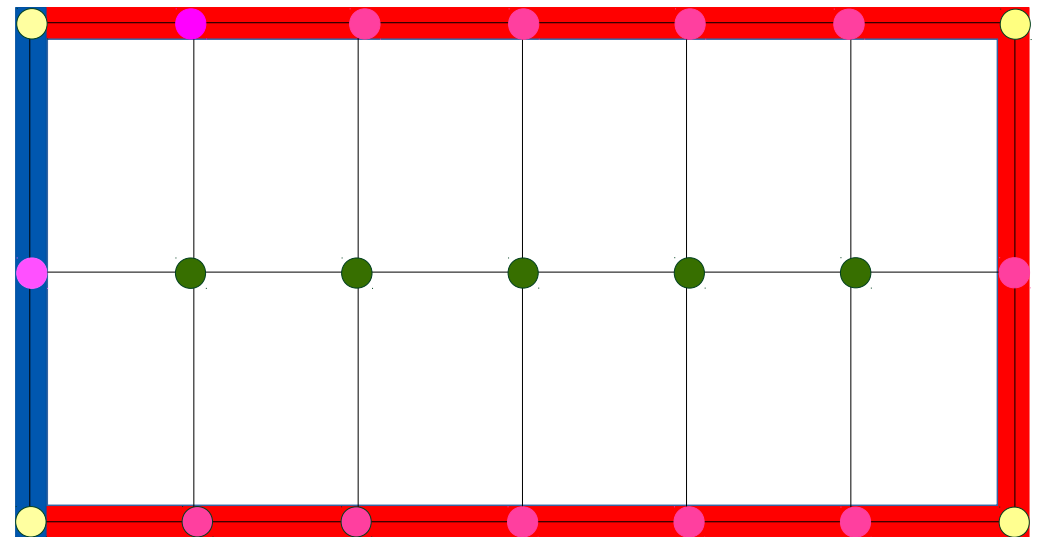
$$f'_n = \frac{f_{n+1} - f_{n-1}}{2h}$$

$$f_{n-1} = f_{n+1} - 2hf'_n$$

$$T_{-1,1} = T_{1,1} - 2h15$$

$$\left. \frac{\partial T(x,y)}{\partial x} \right|_{x=0} = -15\text{C/cm}$$

● Punto auxiliar



Poisson + Neumann

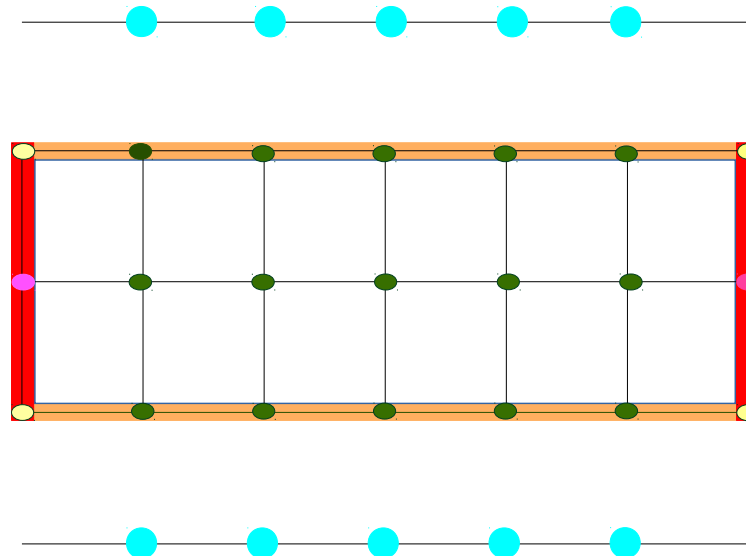
Veamos un ejemplo, siguiendo el Gerald/Wheatly, de ecuación de Poisson con condiciones de Neumann.

Una placa de 8x4 cm, 1cm de grosor, calentada homogéneamente con, $Q \text{ cal / (s cm}^3\text{)}$. La placa tiene una conductividad de $k=0.16 \text{ cal / (s cm}^2\text{) } ^\circ\text{C/cm}$. Consideraremos $Q=5$ u otros valores. Tenemos en este caso la ecuación de Poisson para el estado estacionario:

$$k \nabla^2 T(x, y) = -Q(x, y)$$

$$Q(x, y) \equiv Q$$
$$\frac{Q}{\kappa} = \frac{5}{0.16} \frac{\text{grados}}{\text{cm}^2}$$

Supongamos que perdemos calor por los bordes superior e inferior



$$\left. \frac{\partial T(x, y)}{\partial y} \right|_{x=0} = -15^\circ\text{C/cm}$$

Poisson + Neumann

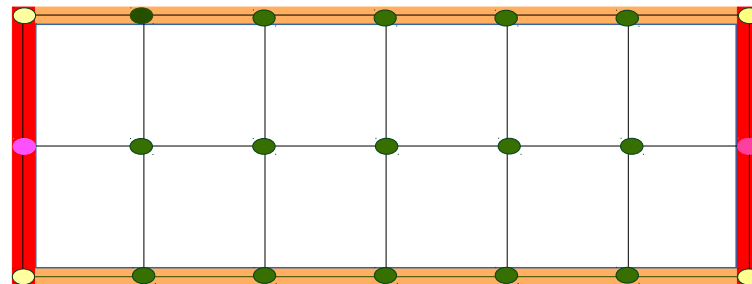
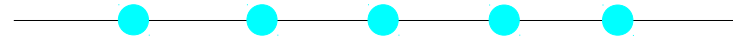
Tendremos el siguiente sistema de ecuaciones,

Para los puntos interiores y frontera de Neumann:

$$T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4T_{i,j} = -h^2 Q/\kappa$$

Utilizando para los puntos auxiliares:

$$T_{i,N_y+1} = T_{i,N_y-1} - 2h15$$



$$T_{i,-1} = T_{i,1} - 2h15$$



Poisson + Neumann

En el caso de la figura tendremos el sistema con:

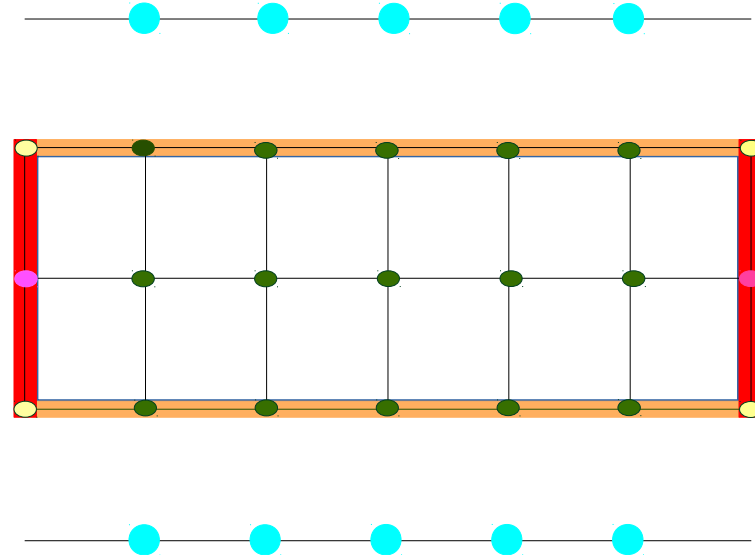
15 ecuaciones de 5 puntos:

$$T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4T_{i,j} = -h^2 Q/\kappa$$

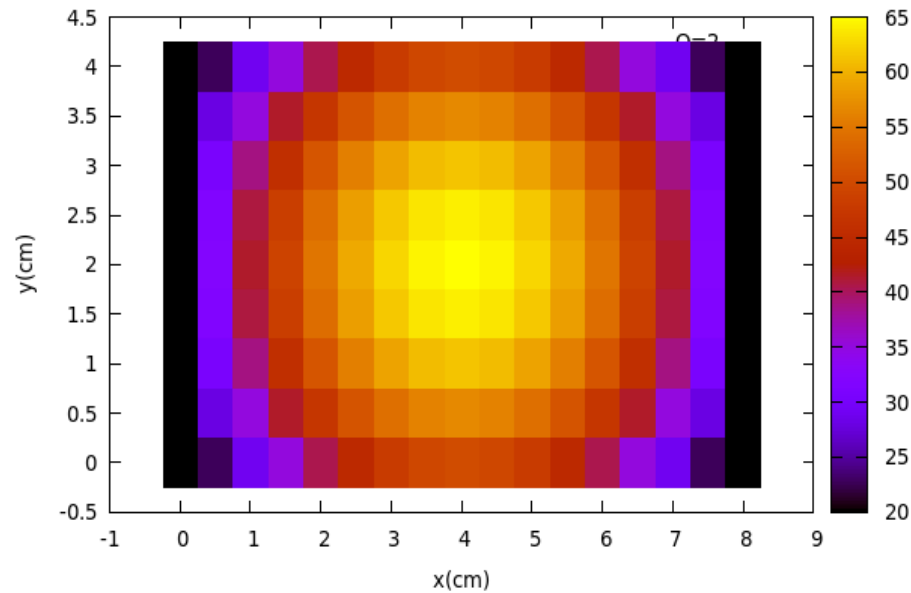
10 ecuaciones para los puntos auxiliares

$$T_{i,N_y+1} = T_{i,N_y-1} - 2h15$$

$$T_{i,-1} = T_{i,1} - 2h15$$

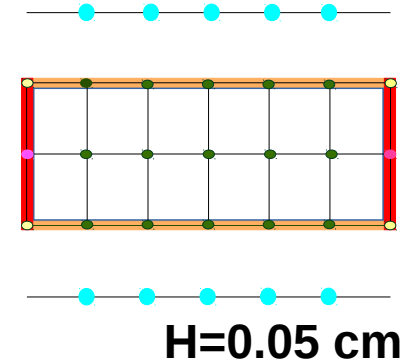
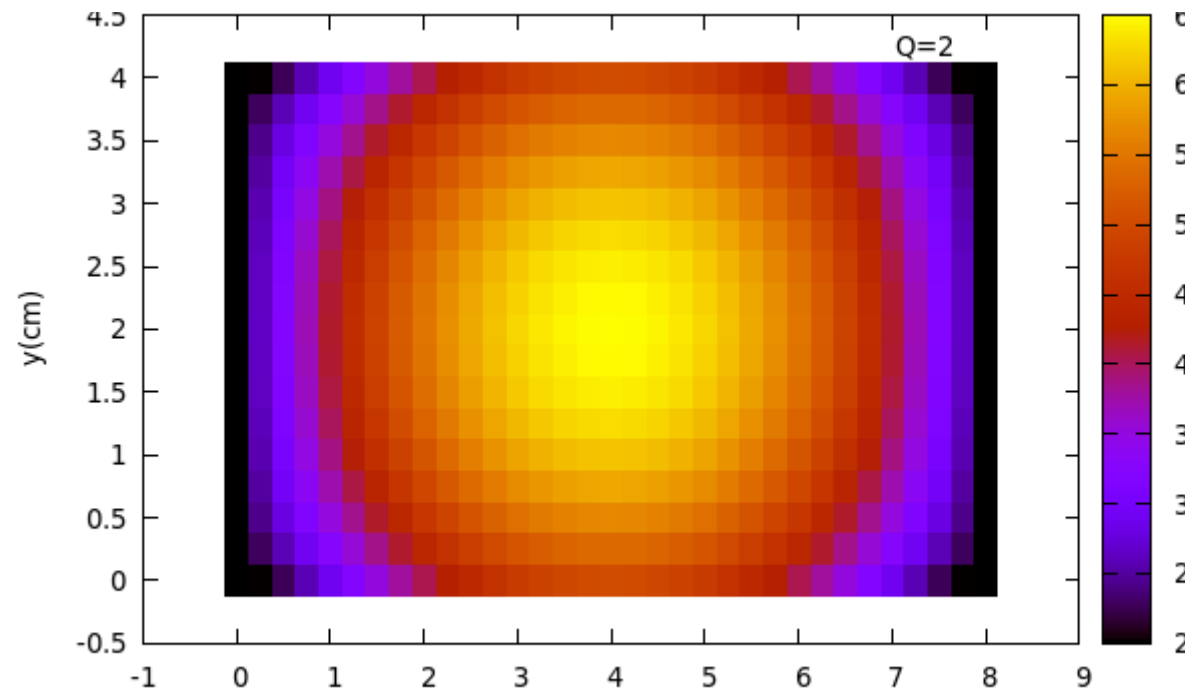


Gauss-Seidel, Q=2



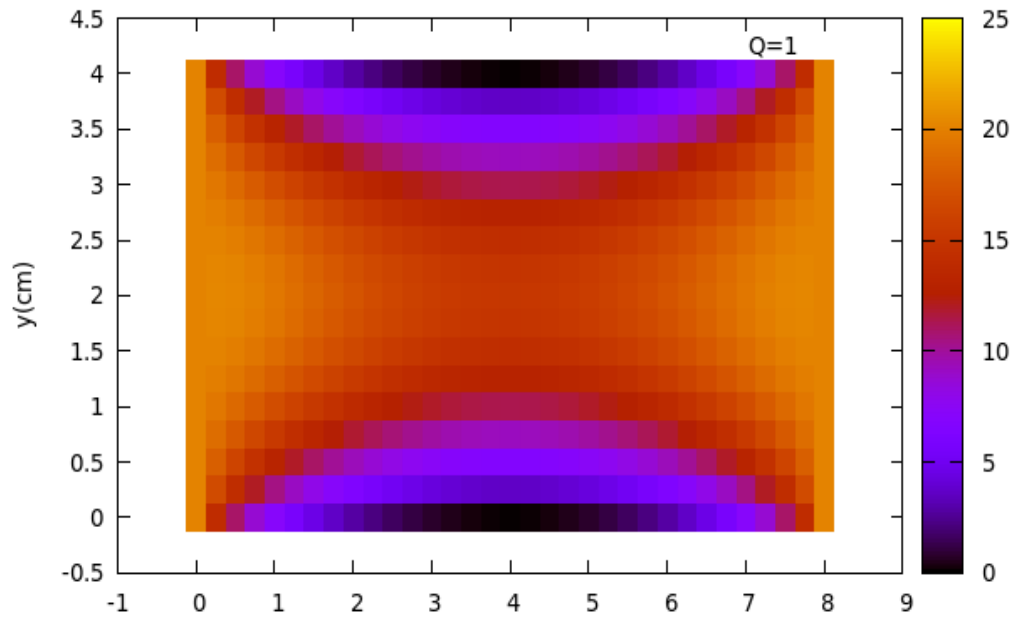
$H=0.1$ cm

Temperatura (grados)



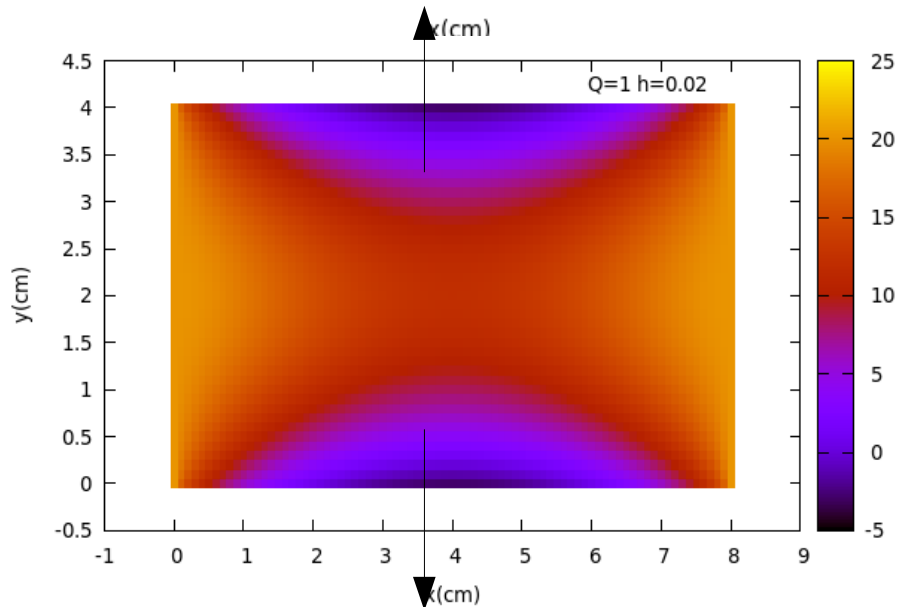
$$T_{i,j}^{\text{nuevo}} = \frac{T_{i+1,j}^{\text{old}} + T_{i-1,j}^{\text{old}} + T_{i,j+1}^{\text{old}} + T_{i,j-1}^{\text{old}} + h^2 Q / \kappa}{4}$$

Gauss-Seidel, $Q=1$

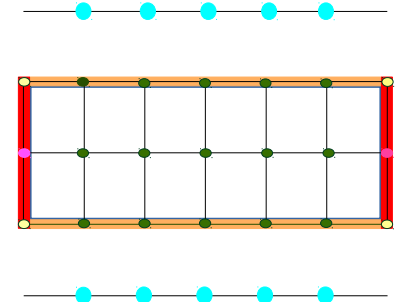


$H=0.05$ cm

Temperatura (grados)



$H=0.02$ cm



Jacobi vs GS vs relaxation

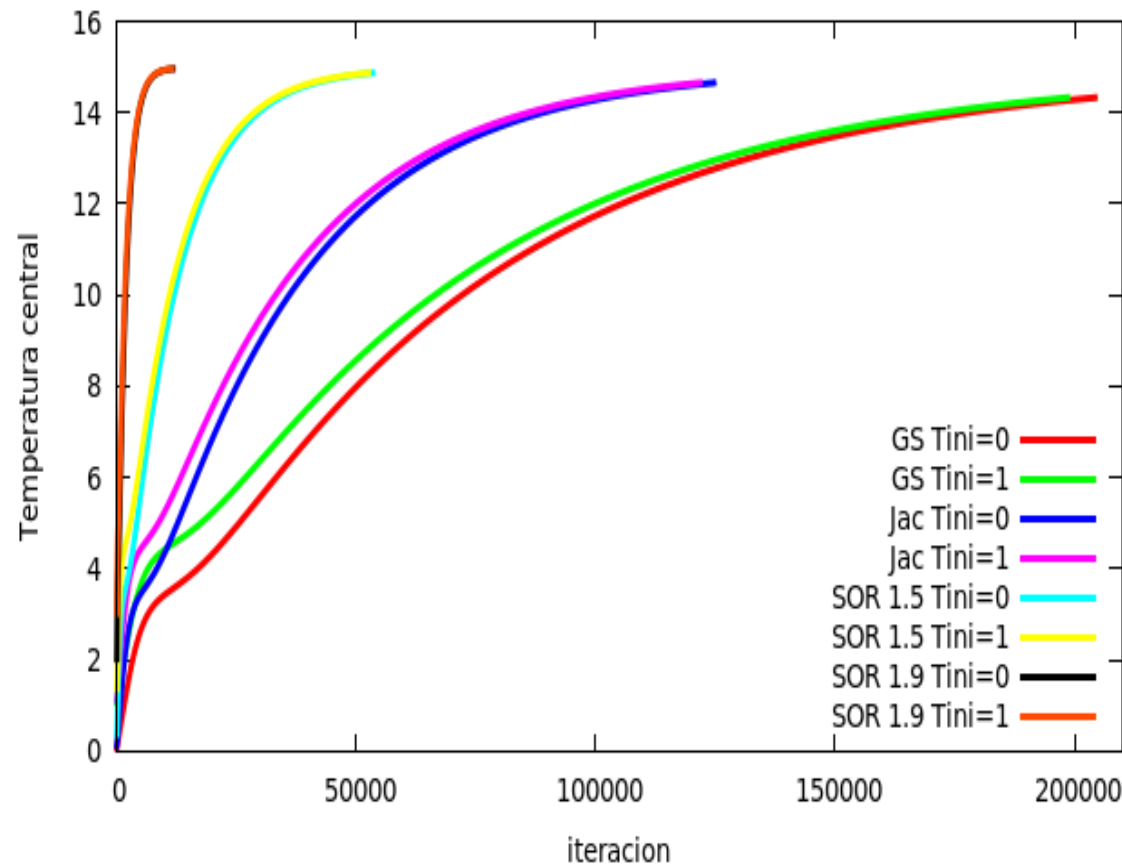
NUMERO DE ITERACIONES
(TOLERANCIA=0.00001)

Condición inicial:
H=0.02

T=1

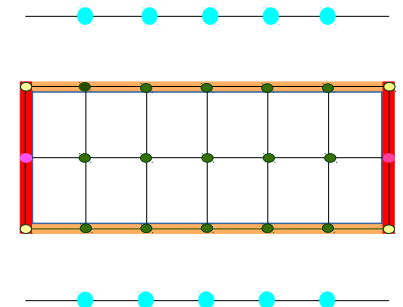
T=0

| | | |
|------------------|--------|--------|
| Gauss-Seidel | 198624 | 204319 |
| Jacobi | 121893 | 124747 |
| SORelajación 1.5 | 52621 | 53576 |
| SORelajacion 1.9 | 11802 | 11645 |

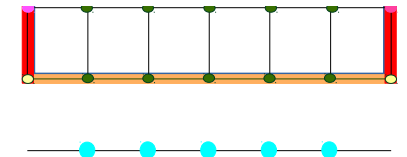
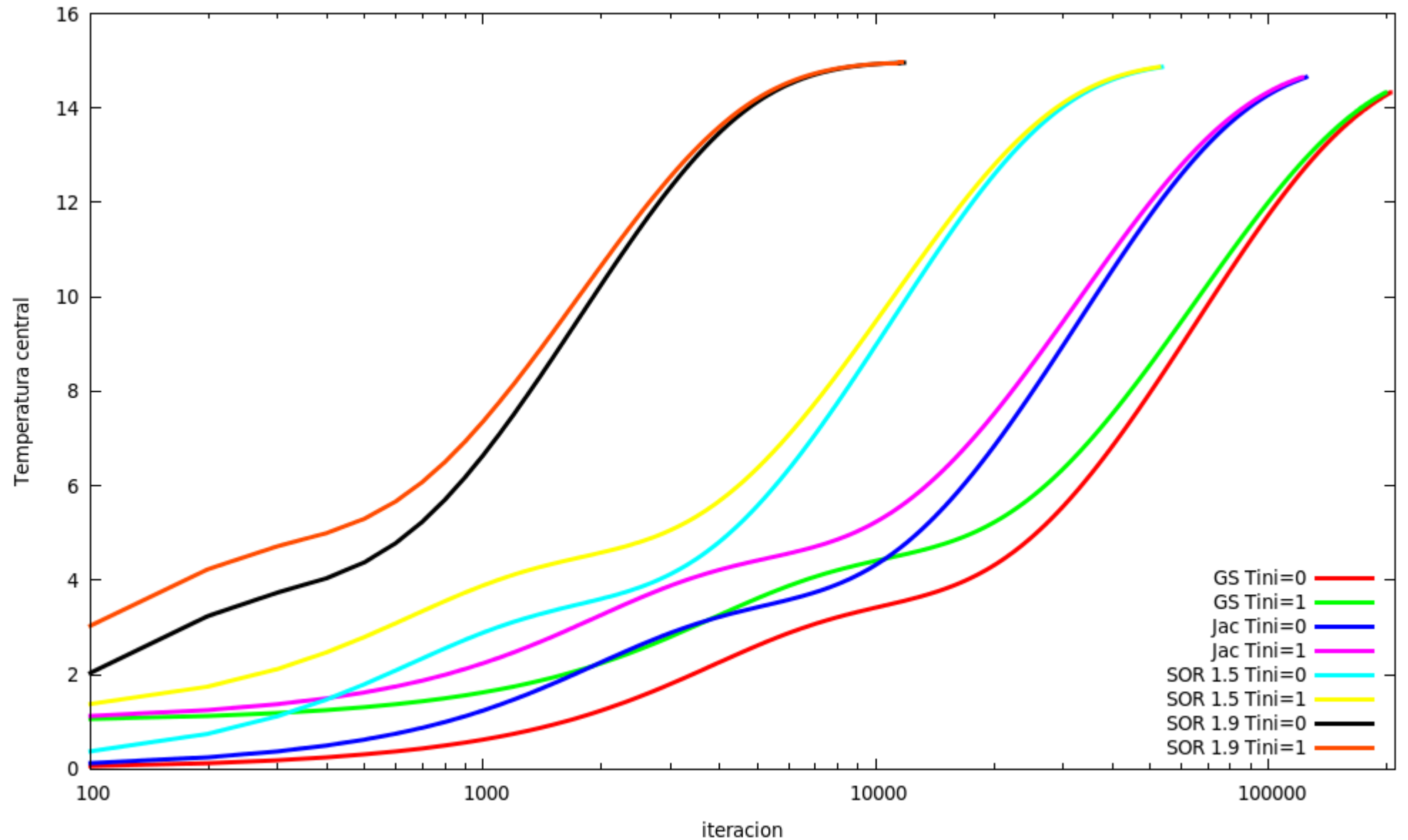


> Más de un factor 10 de mejora utilizando métodos de sobrerrelajación

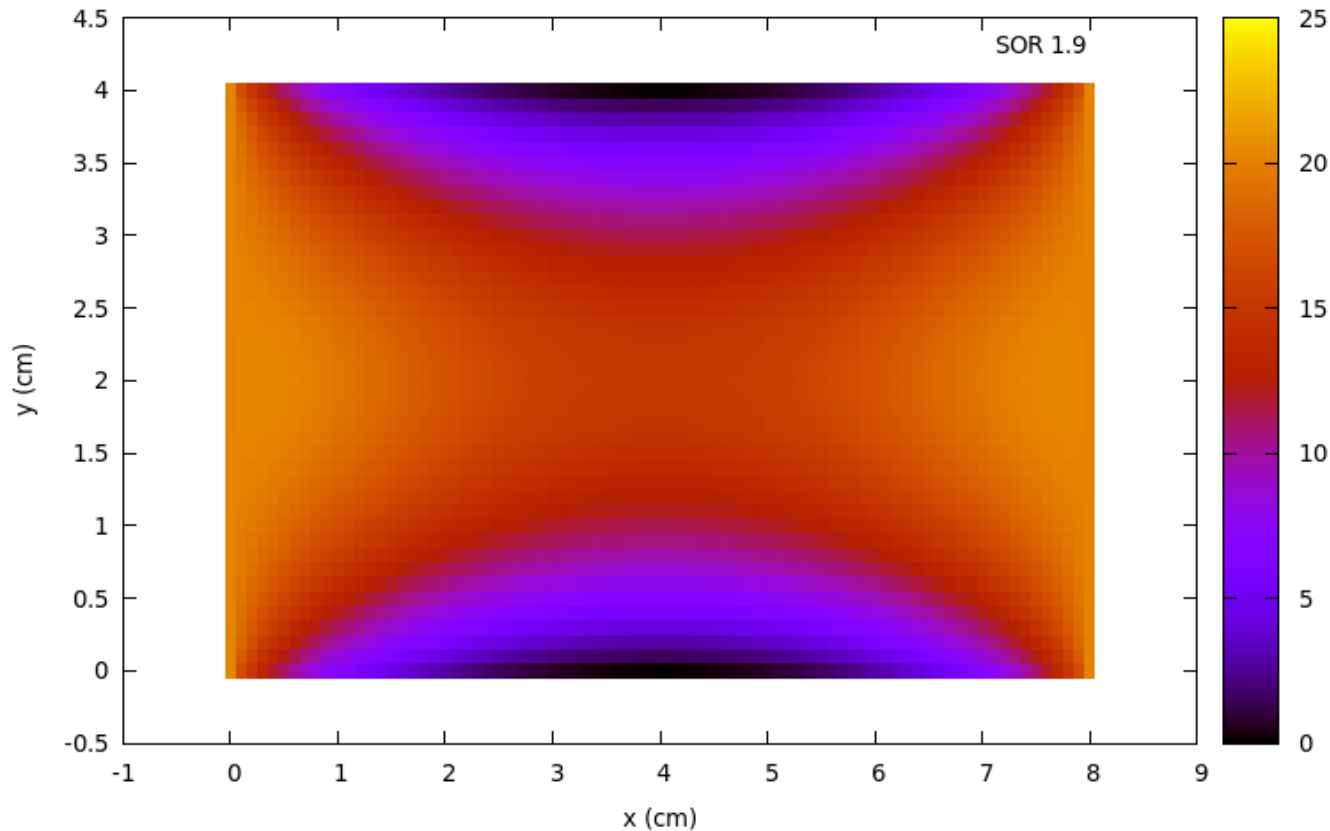
> Dependencia menor en las condiciones iniciales



Jacobi vs GS vs relaxation



Jacobi vs GS vs relaxation (Q=1)

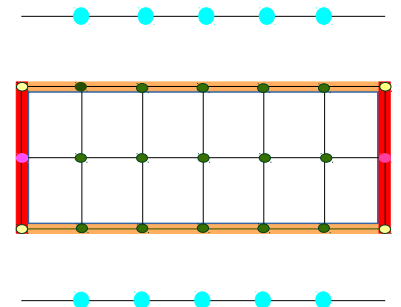


Para hacer esta representación con gnuplot. Un “fichero.dat”

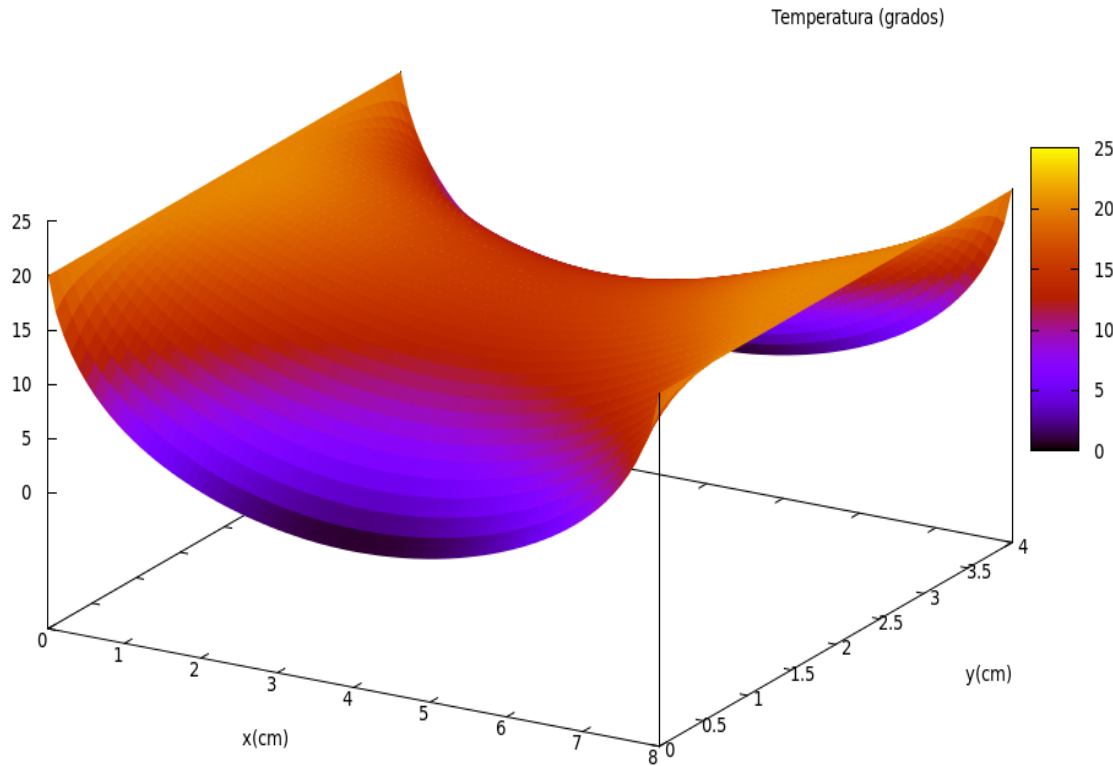
```
#X      Y      TEMPERATURA
X1      y1      T11
X1      y2      T12
.
.
X1      yNx      T1N
Línea en blanco
X2      y1      T11
X2      y2      T12
.
.
X2      yNx      T1N
.
```

Comando:

`plot “fichero.dat” w image`



Jacobi vs GS vs relaxation (Q=1)

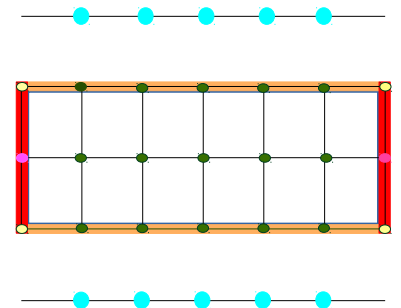


Para hacer esta representación con gnuplot. Un “fichero.dat”

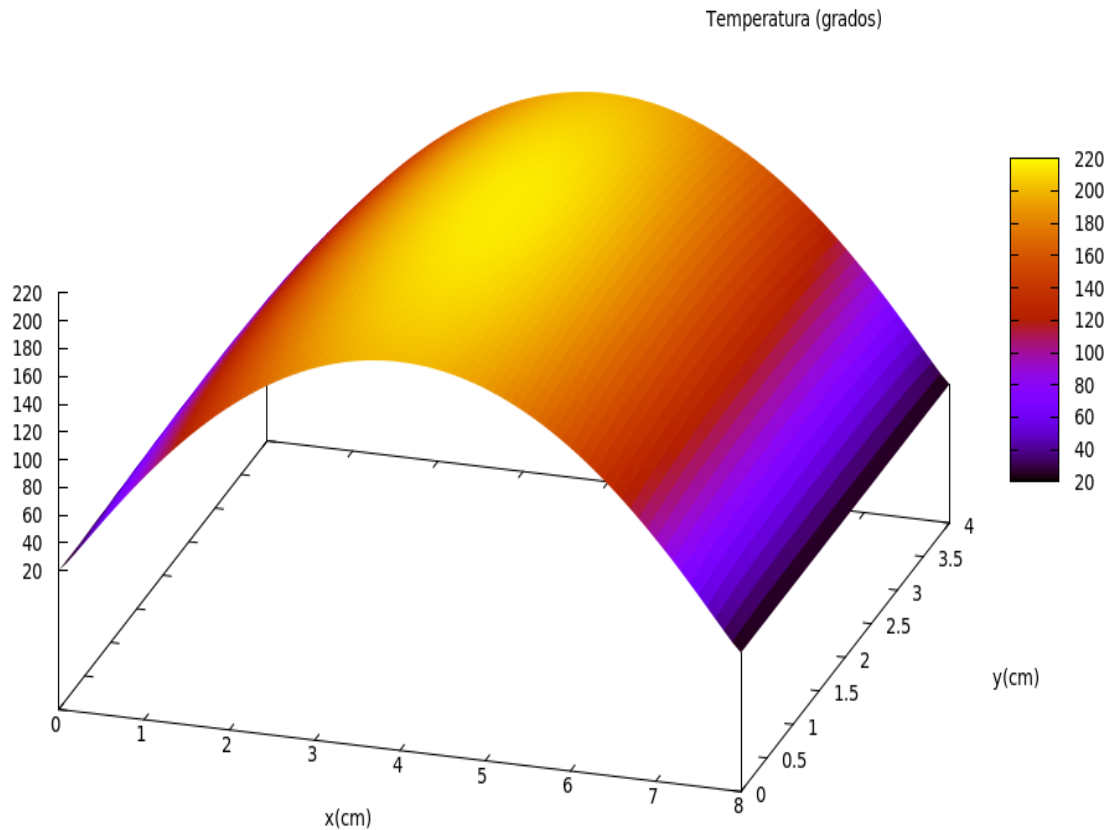
```
#X      Y      TEMPERATURA
X1      y1      T11
X1      y2      T12
.
.
X1      yNx     T1N
Linea en blanco
X2      y1      T11
X2      y2      T12
.
.
X2      yNx     T1N
.
```

Comando:

`sp “fichero.dat” w pm3d`



Jacobi vs GS vs relaxation (Q=5)



sp “fichero.dat” w pm3d

Para hacer esta representación con gnuplot. Un “fichero.dat”

```
#X      Y      TEMPERATURA
X1      y1      T11
X1      y2      T12
.
.
X1      yNx      T1N
Linea en blanco
X2      y1      T11
X2      y2      T12
.
.
X2      yNx      T1N
.
```

