Exercício Programa I

MAP3121 - Métodos Numéricos e Aplicações - 1º Semestre 2020

Nome	NUsp
Isabella Bologna Salomão	9267161
Renato de Oliveira Freitas	9837708

São Paulo, 22 de Maio de 2020

Primeira tarefa

a) **Diferenças Finitas**

Dados:

Fonte de calor: $f(t,x) = 10\cos(10t)x^2(1-x)^2 - (1+\sin(10t))(12x^2 - 12x + 2)$

Solução exata: $u(t,x) = (1+\sin(10t))\,x^2\,(1-x^2)$

Condição inicial: $u0(x) = x^2\,(1-x)^2$

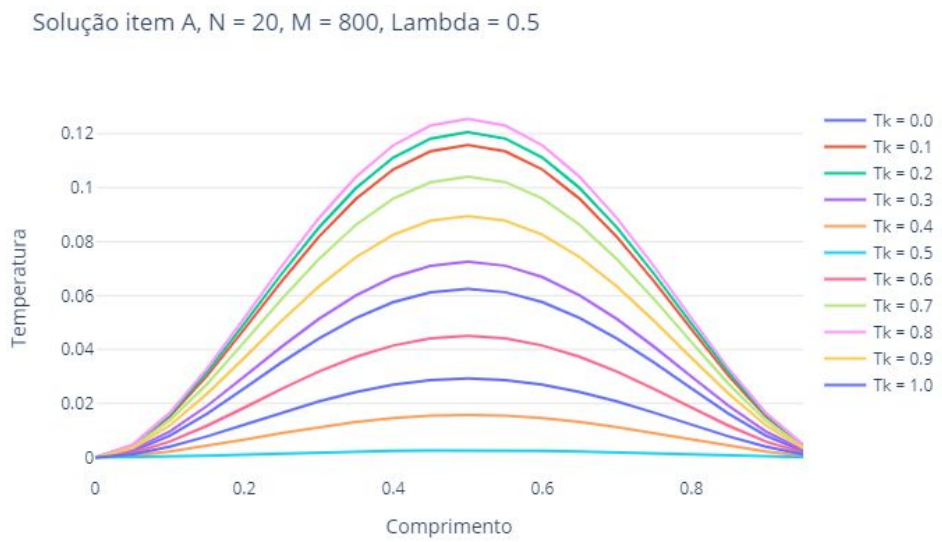
Condição de fronteira: $g1(tk) = 0 \quad g2(tk) = 0$

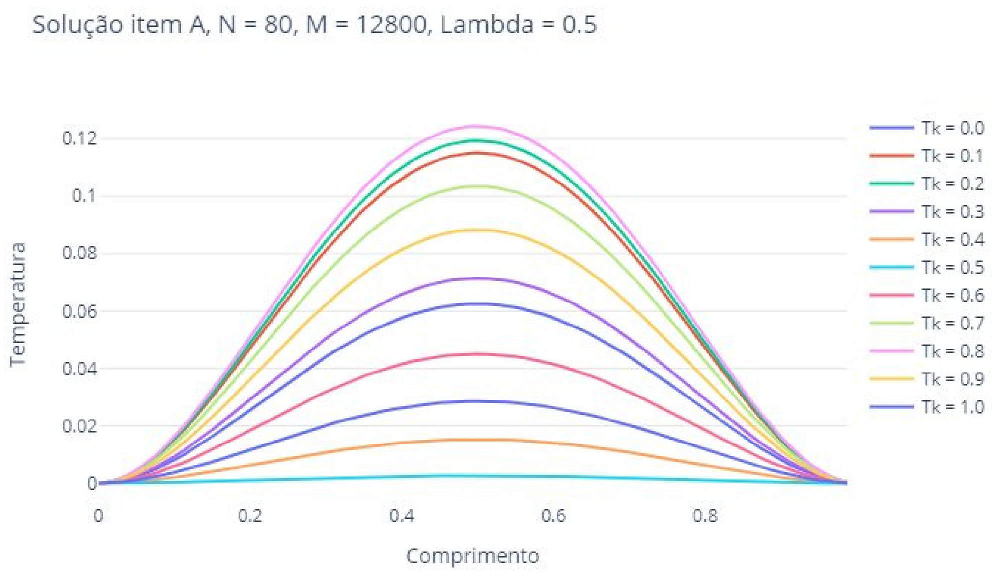
T = 1

Integrações:

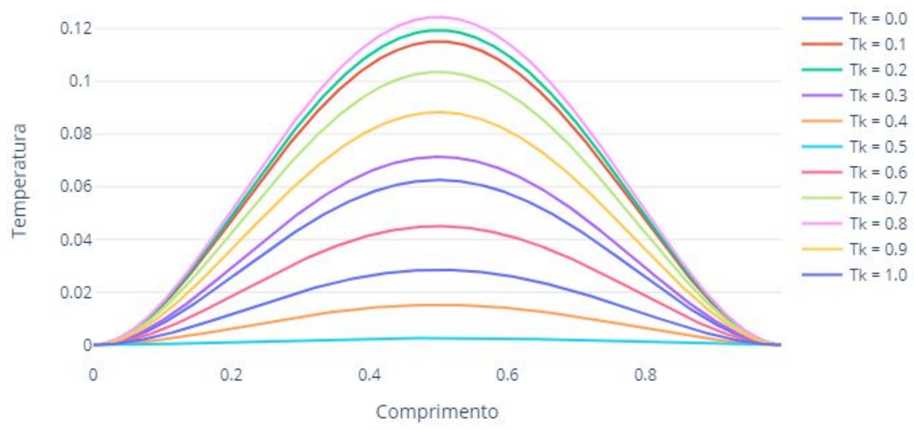
N = 10, 20, 40, 80, 160 e 320

λ = 0.5 e λ = 0.25

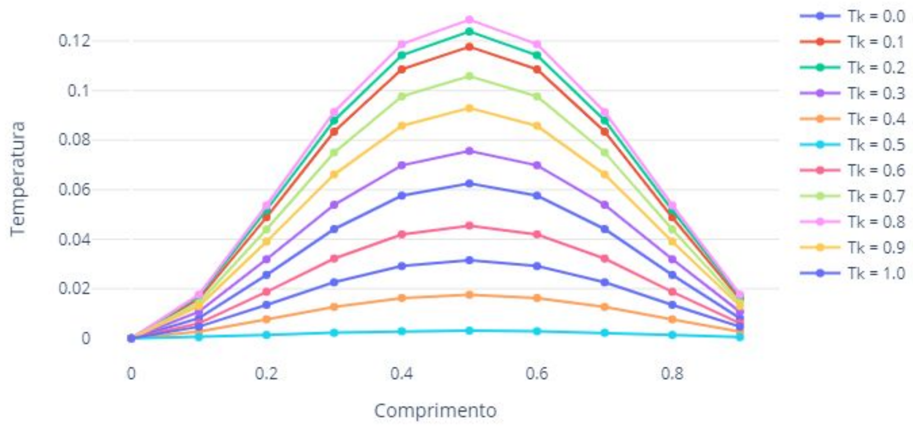




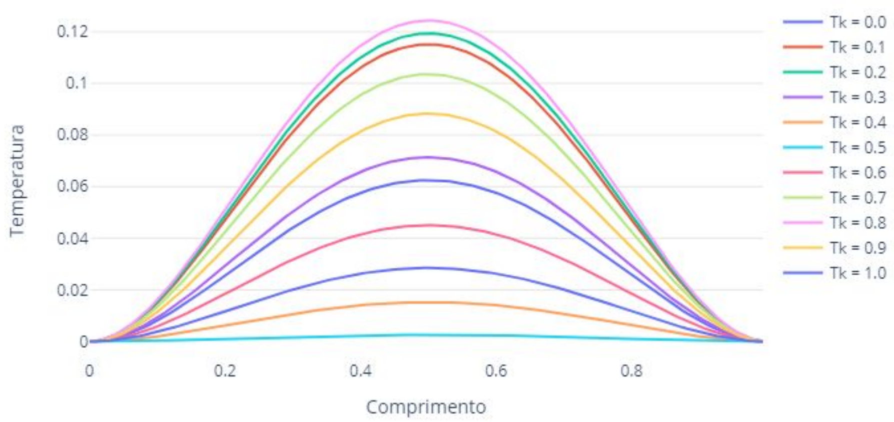
Solução item A, N = 320, M = 204800, Lambda = 0.5

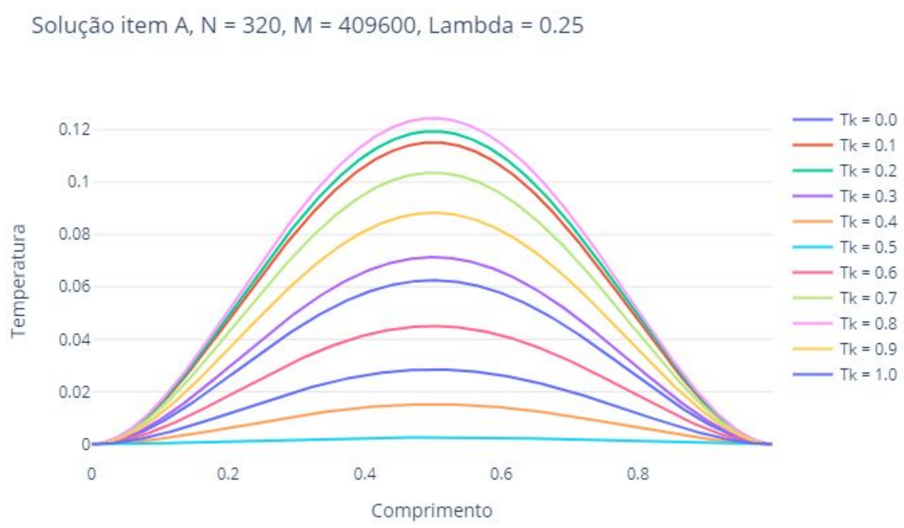


Solução item A, N = 10, M = 400, Lambda = 0.25



Solução item A, N = 160, M = 102400, Lambda = 0.25

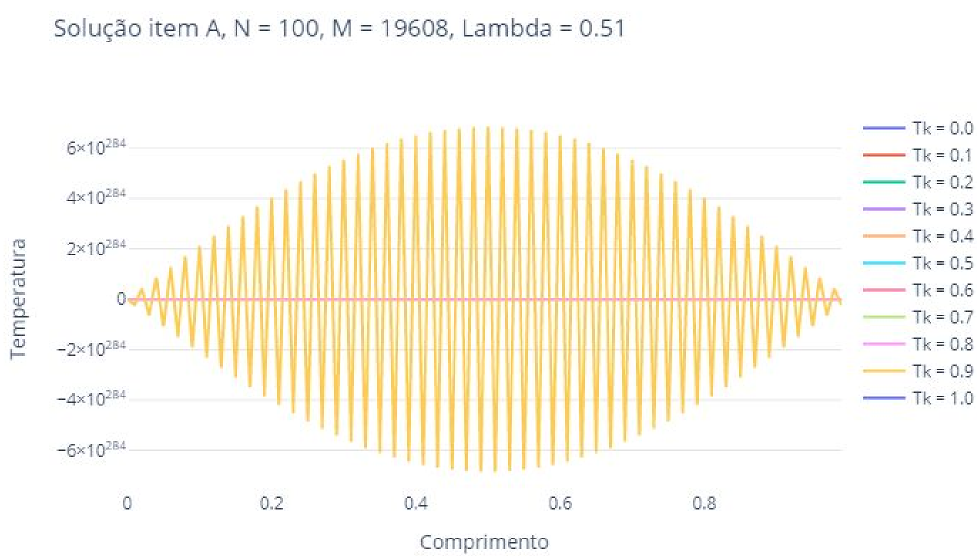




- **λ = 0.51**

Usando λ = 0.51 a condição de convergência usada nesse método, dada pela equação (27), é violada, de forma que o método fica instável.

Equação (27) $\lambda = \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}$



- **Número de Passos**

O número de passos necessários para o cálculo será $M \geq \frac{T.N^2}{\lambda}$, $M \in Z$

N	λ = 0.5	λ = 0.25
640	819200	1638400
1280	3276800	6553600

b) Determine quem deve ser $u0(x)$, $g1(t)$, $g2(t)$ e $f(t, x)$ de forma que a solução exata seja dada por $u(t,x) = e^{t-x}\, cos(5tx)$

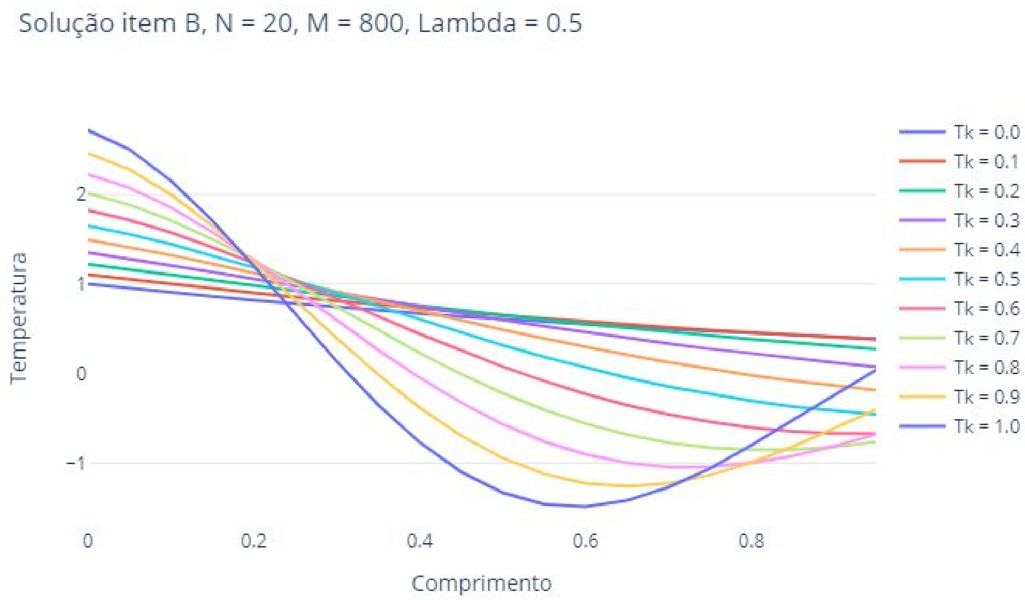
$u0(x) = u(0,x) = e^{0-x}\, cos(5.0.x) = e^{-x}$

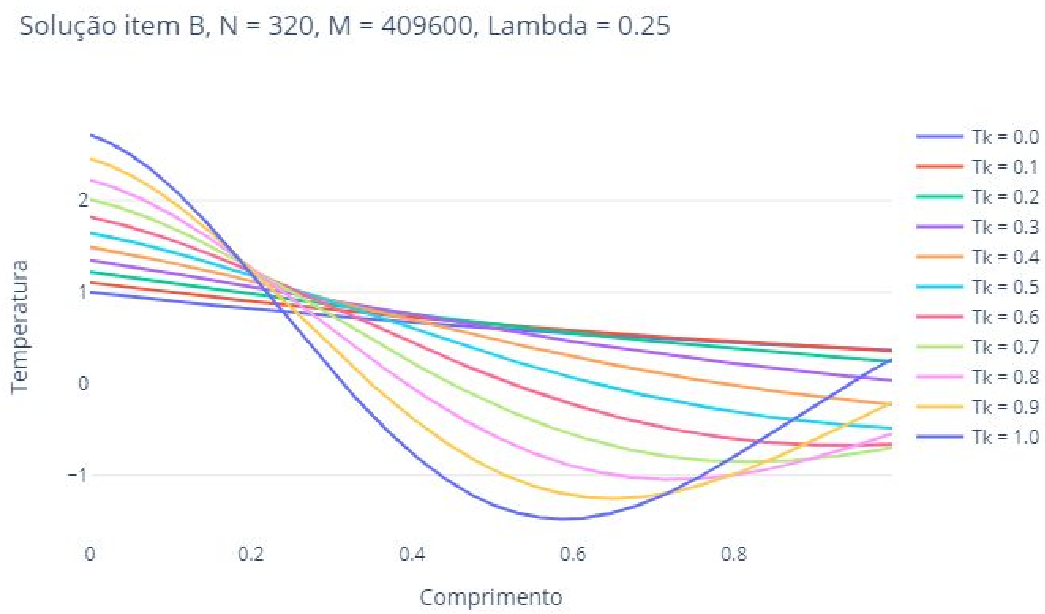
$g1(t) = u(t,0) = e^{t-0}\, cos(5t.0) = e^{t}$

$g2(t) = u(t,1) = e^{t-1}\, cos(5t)$

$u_t(t,x) = u_{xx}(t,x) + f(t,x) \rightarrow f(t,x) = u_t(t,x) - u_{xx}(t,x)$
$u_t(t,x) = e^{t-x}\,[cos(5tx) - 5x\, sin(5tx)]$
$u_{xx}(t,x) = e^{t-x}\,[(1-25t^2)\, cos(5tx) + 10t\, sin(5tx)]$

$f(t,x) = e^{t-x}\,[(25t^2\, cos(5tx) - 5(x+2t)\, sin(5tx)]$





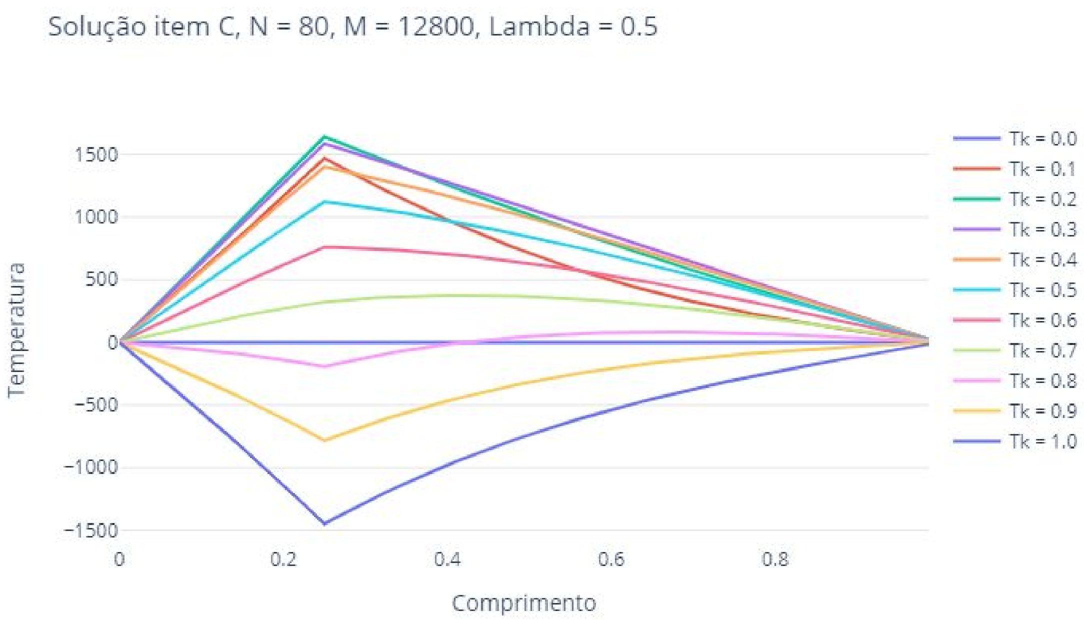
- **Erro máximo em T=1**

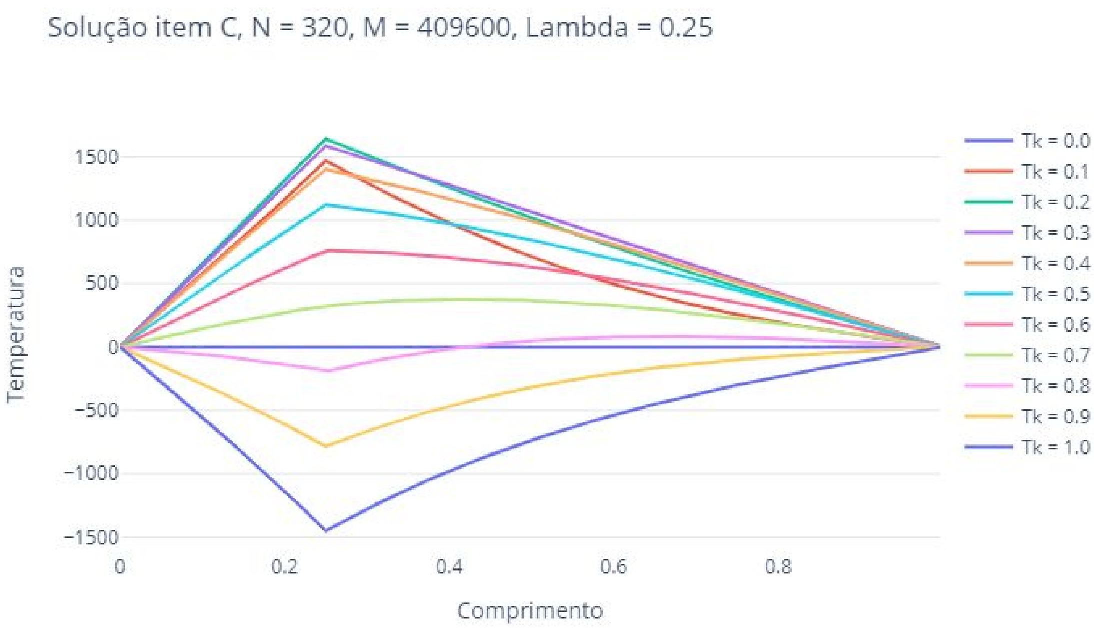
Comparação do erro para cada N e λ nos dois itens utilizando diferenças finitas

	Item A		Item B	
N	λ = 0.5	λ = 0.25	λ = 0.5	λ = 0.25
10	3.16E-03	3.05E-03	5.05E-02	5.00E-02
20	7.84E-04	7.58E-04	1.26E-02	1.25E-02
40	1.96E-04	1.89E-04	3.15E-03	3.13E-03
80	4.89E-05	4.73E-05	7.88E-04	7.82E-04
160	1.22E-05	1.18E-05	1.97E-04	1.95E-04
320	3.05E-06	2.96E-06	4.93E-05	4.89E-05

Ao dobrarmos o N, vemos que o erro, em ambos os casos diminui em 4 vezes. Isto é, o erro diminui em uma relação de segunda ordem.

c) Dados:
T = 1
Condição Inicial: 0
fonte pontual: $r(t) = 10000\,(1 - 2\,t^2\,)$
$gh(x) = 1/h\,,\ se\ p - h/2 \leq x \leq p + h/2\,,\ e\ gh(x) = 0\ caso\ contrário$
fonte em p = 0.25
Condição de fronteira: $g1(t) = g2(t) = 0$





Segunda tarefa

a) Procedimento para calcular a solução *x* de $Ax = LDL^t x = b$

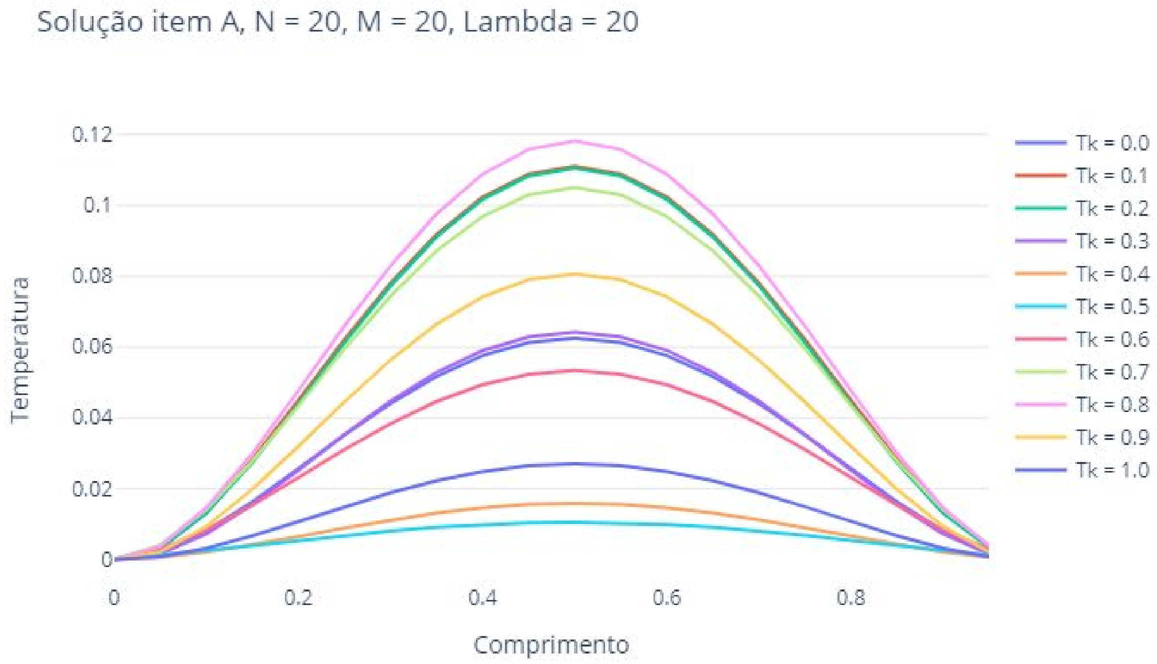
```
def create_A_matrix(l, size):
    '''Creates a representation of A with two arrays

    Args:
    -----
        l:    Lambda value
        size: Size of one dimension of A

    Returns:
    --------
        Matrix A represented as two arrays (A, B)
    '''

    A = np.array([1 + 2*l] * size, dtype=float)
    B = np.array([-l] * size, dtype=float)

    B[0] = 0  # valor l1 = 0

    return A, B
```

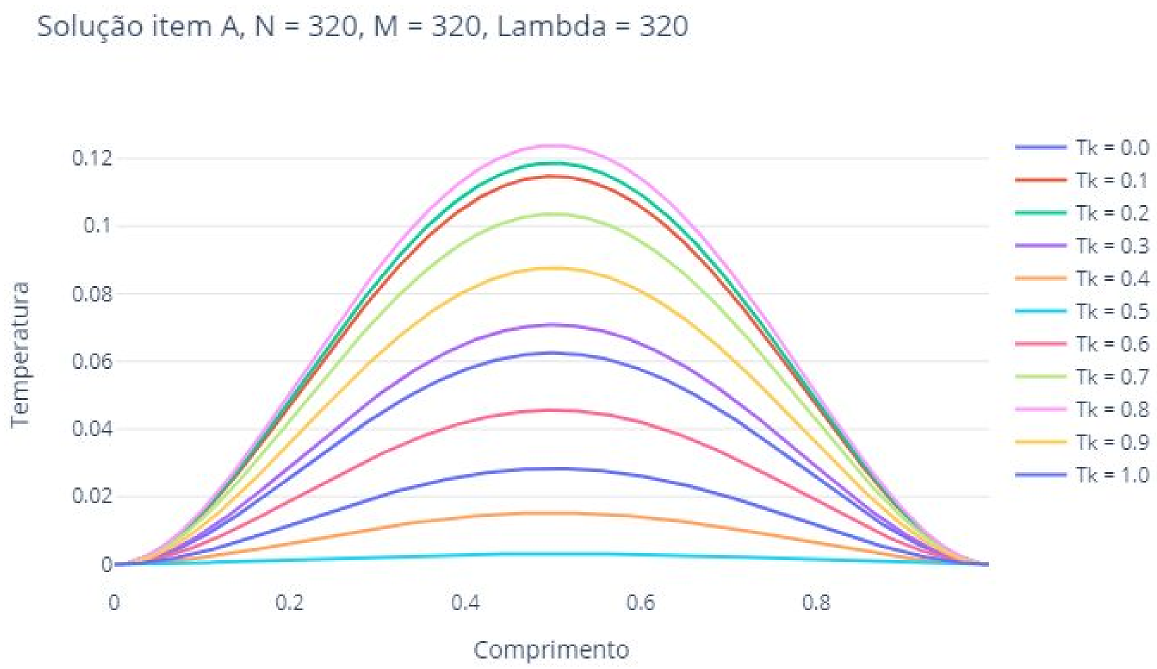
```python
def LDL_decomposition(A, B):
    '''LDLT decomposition of a matrix represented by two arrays

    Args:
    -----
        A:  diagonal of the original matrix
        B:  subdiagonal of the original matrix

    Returns:
    -------
        LDL decomposition represented as two arrays (L, D)
    '''

    L = np.array([0] * len(A), dtype=float)
    D = np.array([0] * len(A), dtype=float)

    D[0] = A[0]
    L[0] = 0

    # Calculo dos valores 'Di' e 'Li' de cada matriz
    for i in range(1, len(A)):
        L[i] = B[i] / D[i - 1]
        D[i] = A[i] - (L[i]**2 * D[i - 1])

    return L, D
```

```python
    # Loop principal da solução do sistema "[L][D][Lt] [x] = [b]" para cada tk,
k=1...M
    for k in range(1, Item.M + 1):
        b = calc_b(k) # calcula o vetor correspondente ao lado direito do sistema

        # Primeiro, resolvemos [L][D] [y] = [b]
        y[0] = b[0] / D[0]
        for i in range(1, Item.N-1):
            y[i] = ((b[i] - L[i] * (y[i-1] * D[i-1])) / D[i])

        # Por ultimo, [Lt] [x] = [y] para achar a solução x
        Item.u[k][Item.N-1] = y[-1]
        for i in range(Item.N-2, 0, -1):
            Item.u[k][i] = y[i-1] - (L[i] * Item.u[k][i+1])
```

b) Método de Euler implícito, utilizando Δt = Δx.

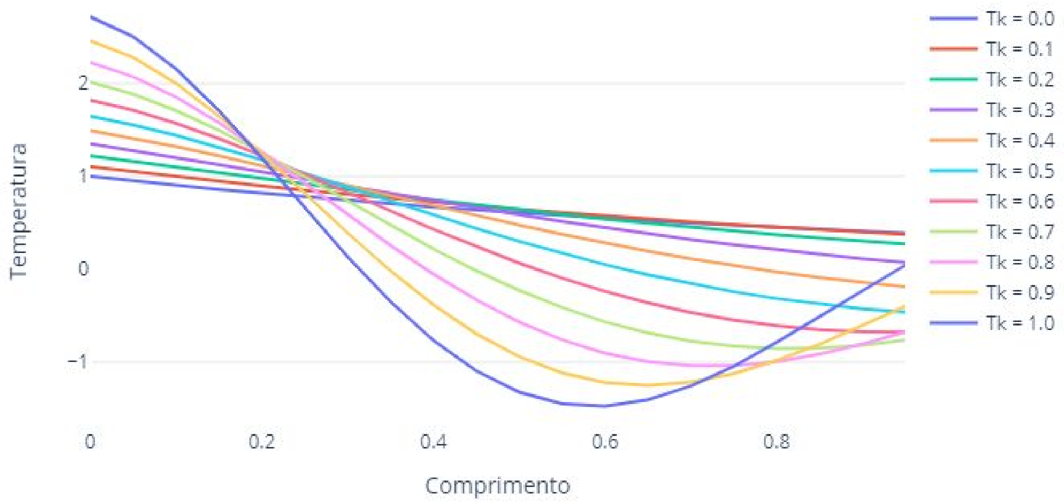
- Item A



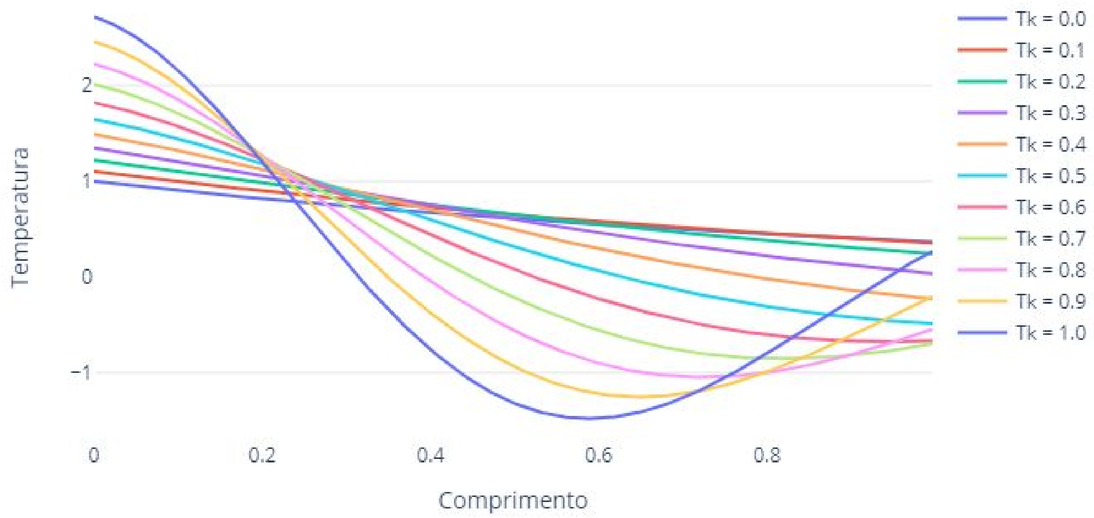


- Item B

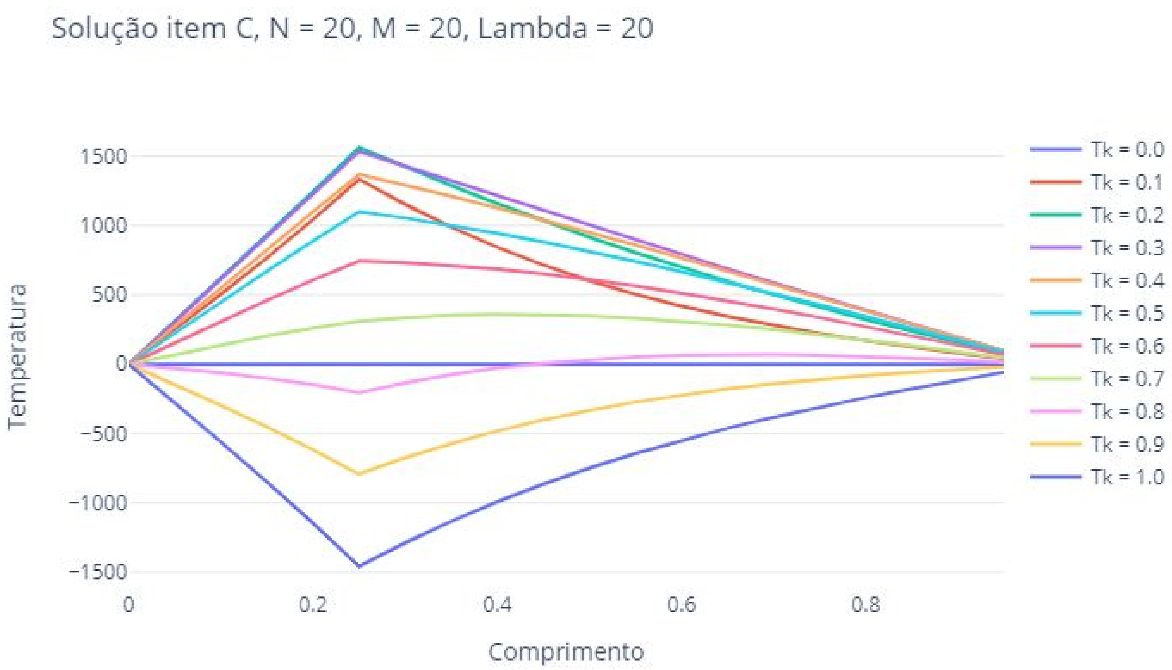
Solução item B, N = 20, M = 20, Lambda = 20



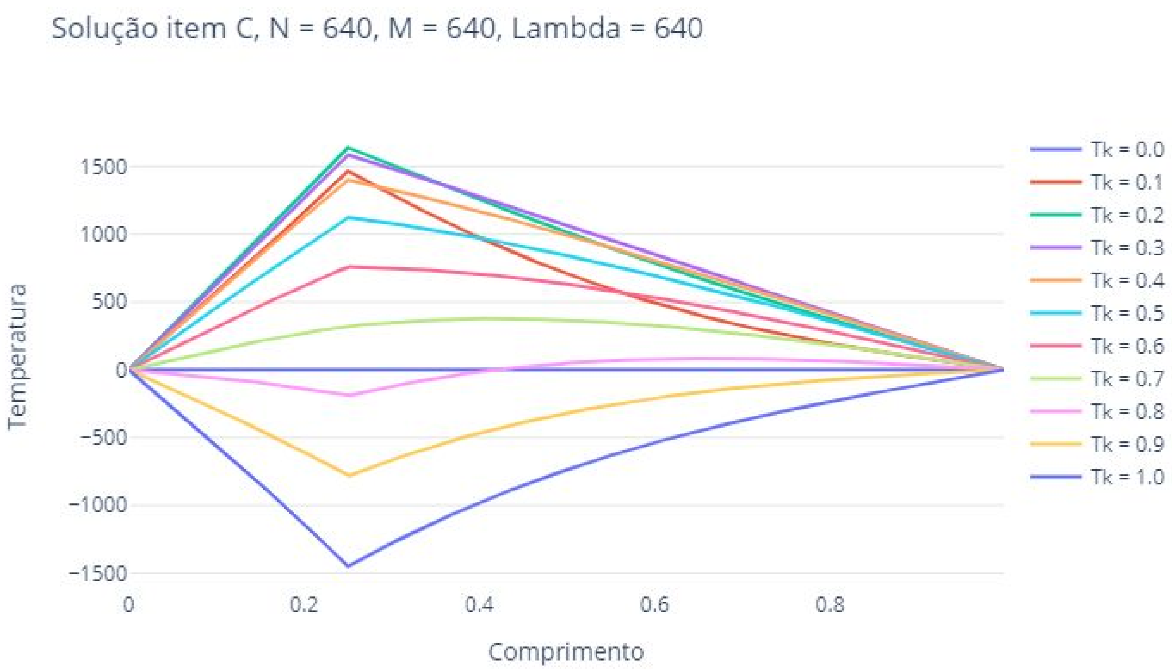
Solução item B, N = 320, M = 320, Lambda = 320



- Item C



Solução item C, N = 20, M = 20, Lambda = 20
1500
1000
500
0
−500
−1000
−1500
0
0.2
0.4
0.6
0.8
Temperatura
Comprimento
Tk = 0.0
Tk = 0.1
Tk = 0.2
Tk = 0.3
Tk = 0.4
Tk = 0.5
Tk = 0.6
Tk = 0.7
Tk = 0.8
Tk = 0.9
Tk = 1.0



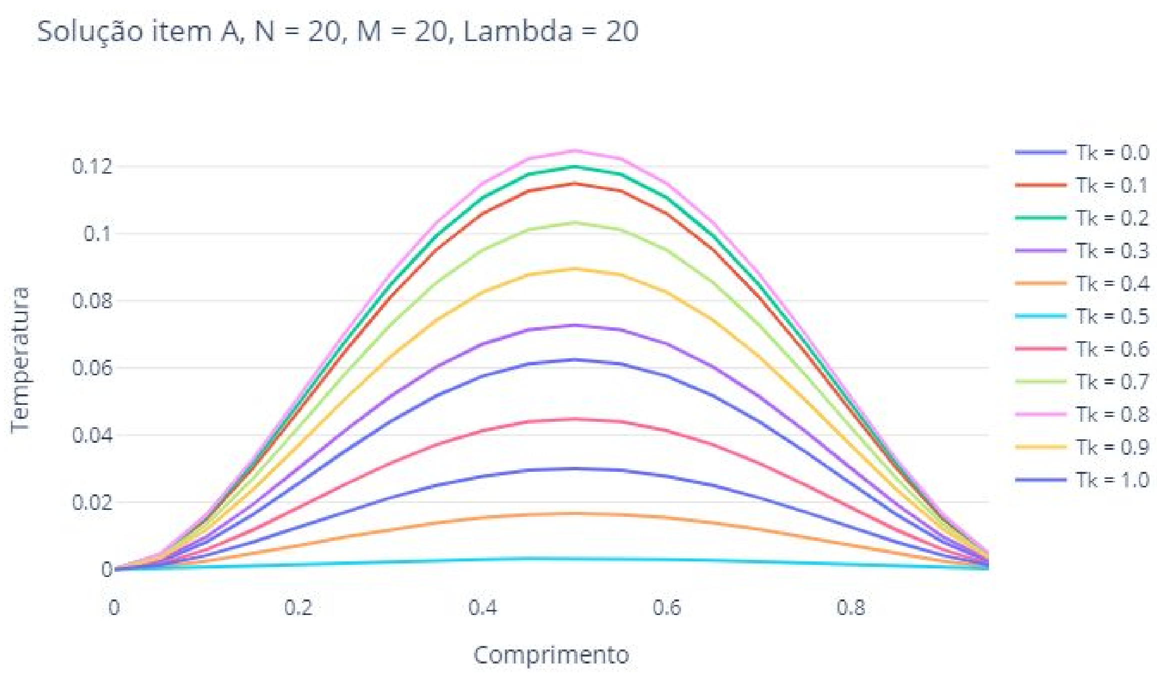
Solução item C, N = 640, M = 640, Lambda = 640
1500
1000
500
0
−500
−1000
−1500
0
0.2
0.4
0.6
0.8
Temperatura
Comprimento
Tk = 0.0
Tk = 0.1
Tk = 0.2
Tk = 0.3
Tk = 0.4
Tk = 0.5
Tk = 0.6
Tk = 0.7
Tk = 0.8
Tk = 0.9
Tk = 1.0

- **Erro máximo em T=1**

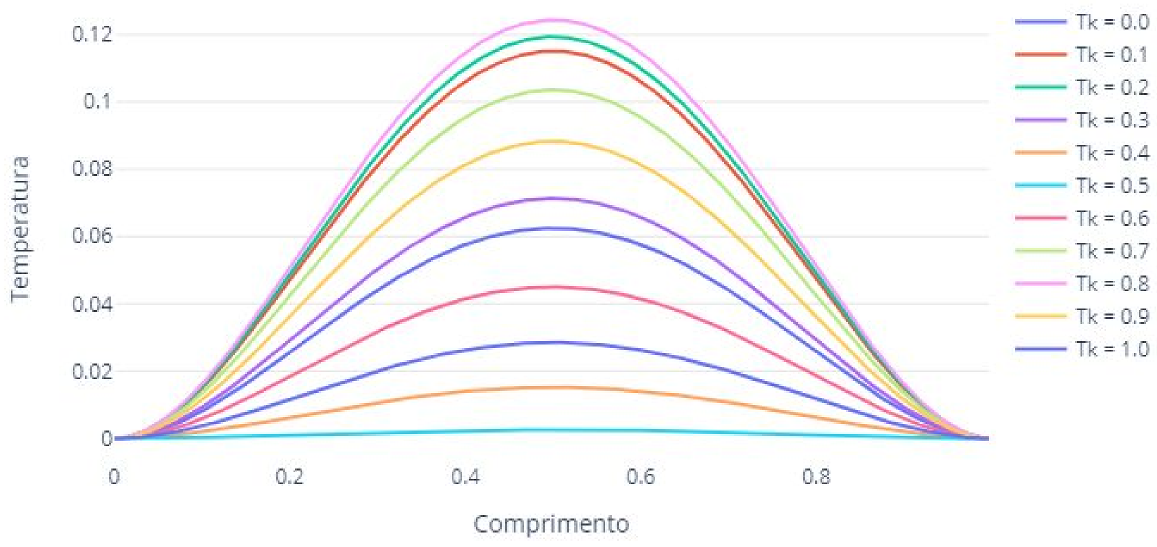
N	Item A	Item B
10	2.08E-03	4.50E-02
20	1.50E-03	9.37E-03
40	8.79E-04	6.28E-03
80	4.74E-04	3.61E-03
160	2.46E-04	1.93E-03
320	1.25E-04	9.97E-04

Vemos para esse método que o erro cai, com o aumento do N, com ordem 1

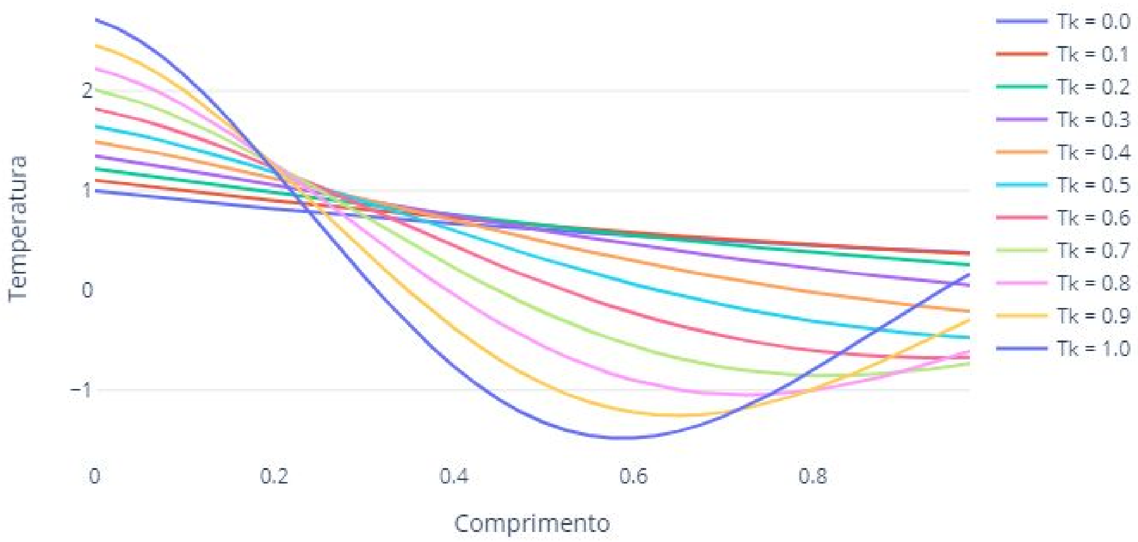
c) Método de Crank-Nicolson.

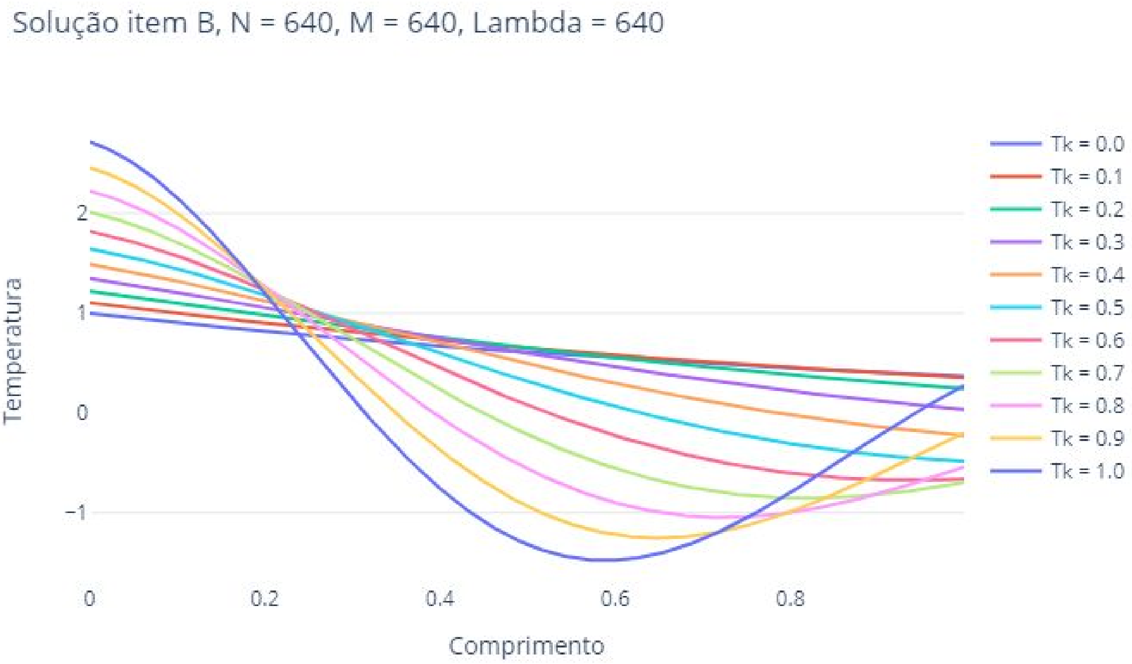


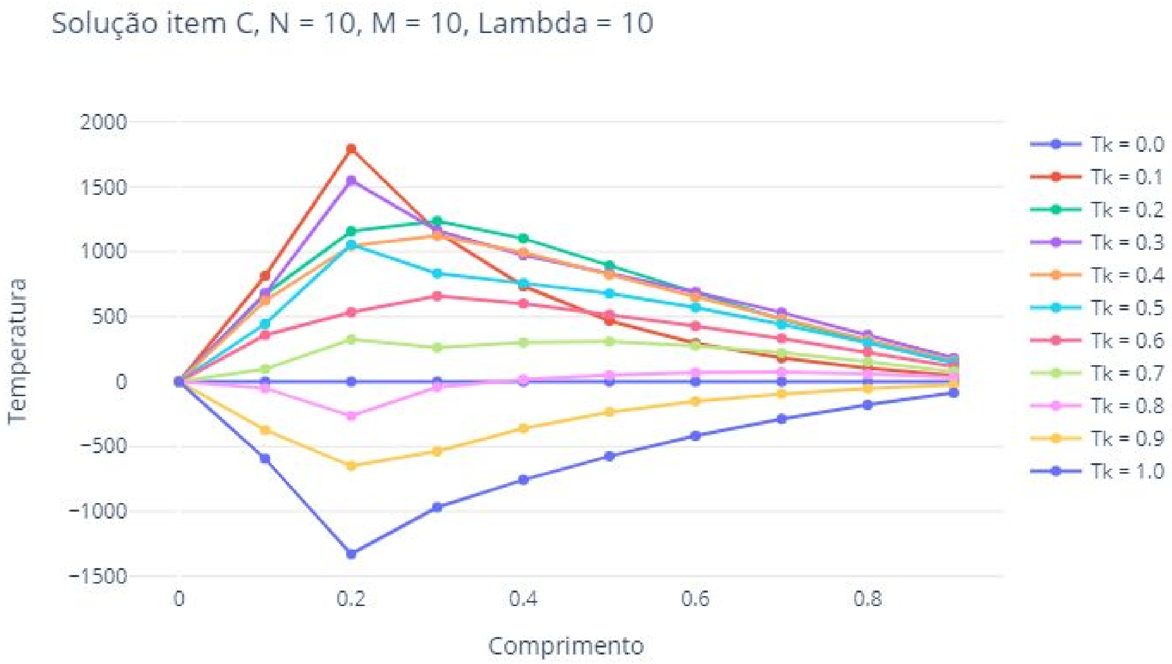
Solução item A, N = 320, M = 320, Lambda = 320

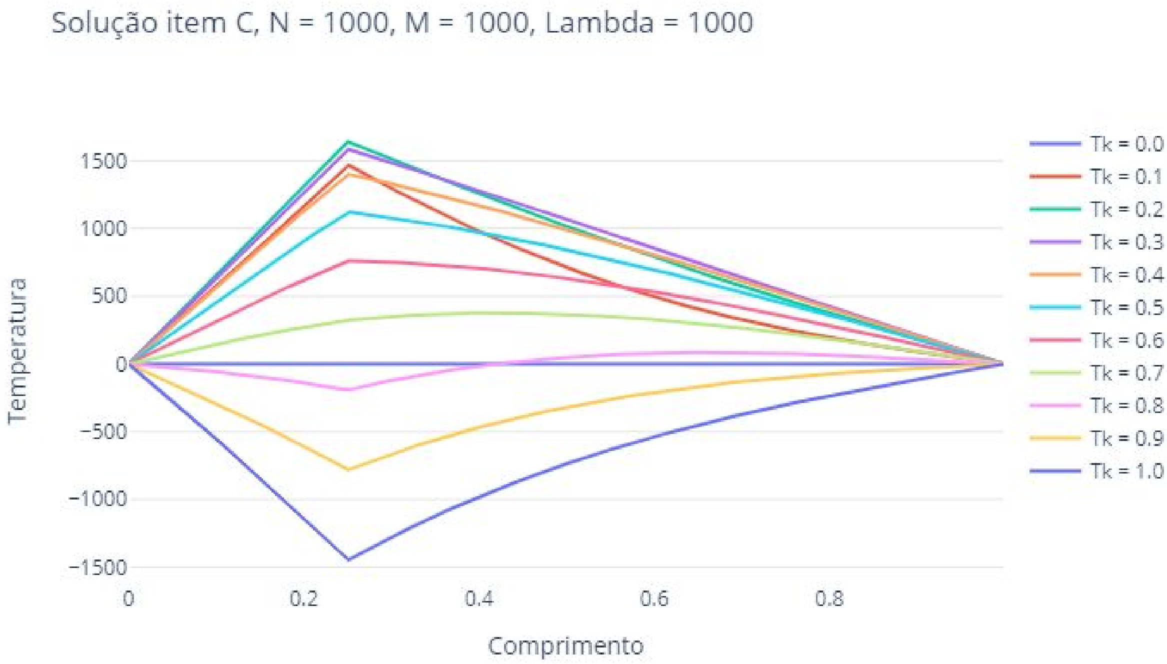


Solução item B, N = 40, M = 40, Lambda = 40




Solução item B, N = 640, M = 640, Lambda = 640
Temperatura
Comprimento
2
1
0
−1
0
0.2
0.4
0.6
0.8
Tk = 0.0
Tk = 0.1
Tk = 0.2
Tk = 0.3
Tk = 0.4
Tk = 0.5
Tk = 0.6
Tk = 0.7
Tk = 0.8
Tk = 0.9
Tk = 1.0


Solução item C, N = 10, M = 10, Lambda = 10
Temperatura
Comprimento
2000
1500
1000
500
0
−500
−1000
−1500
0
0.2
0.4
0.6
0.8
Tk = 0.0
Tk = 0.1
Tk = 0.2
Tk = 0.3
Tk = 0.4
Tk = 0.5
Tk = 0.6
Tk = 0.7
Tk = 0.8
Tk = 0.9
Tk = 1.0


Solução item C, N = 1000, M = 1000, Lambda = 1000
Temperatura
Comprimento
1500
1000
500
0
−500
−1000
−1500
0
0.2
0.4
0.6
0.8
Tk = 0.0
Tk = 0.1
Tk = 0.2
Tk = 0.3
Tk = 0.4
Tk = 0.5
Tk = 0.6
Tk = 0.7
Tk = 0.8
Tk = 0.9
Tk = 1.0