INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
UNIVERSIDADE DE SÃO PAULO

Bacharelado em Matemática Aplicada e Computacional

Métodos Numéricos em Equações Diferenciais II – EP 1

Prof. Sergio Oliva

Tiago Fernandes D'Agostino N º USP 3759732

São Paulo

2º semestre de 2018

Sumário

Modelo e Problema

Considere uma barra uniforme de comprimento l com temperatura não uniforme deitada no eixo x, de $x = 0$ até $x = l$. Assumimos que os lados da barra são isolados e somente os extremos podem ser expostos. Além, assumimos que existe um termo fonte de calor na barra definido pela função $f(t, x; \kappa)$. Assim, obtemos um modelo parabólico não homogêneo com condições de Dirichlet homogênea nos extremos e dado inicial abaixo, onde u é a função estado e $k = (k_1, k_2)$.

$$\partial_t u(t,x) = \partial_{xx}^2 u(t,x) + f(t,x,k), \qquad \text{para cada } x \in (0,l) \; e \; t > 0$$

$$u(t,0) = u(t,l) = 0, \qquad \text{para cata t > 0}$$

$$u(0,x) = g(x) \qquad \text{para cada } x \in (0,l)$$

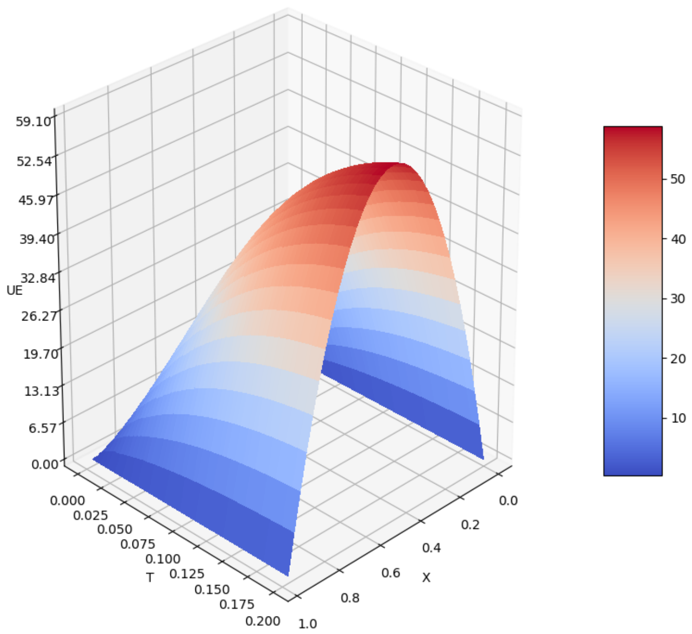
Tarefa Geral

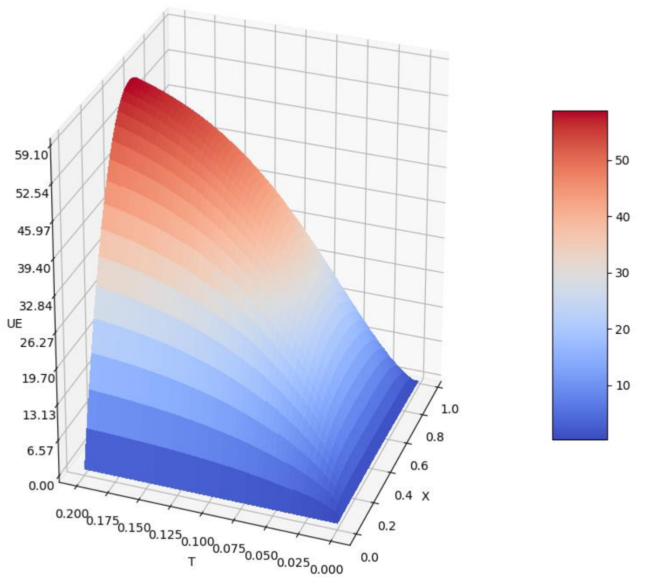
Para esse problema, usamos os dados:

l =1	
$f(t,x,k) = k_1 e^{-k_2 t}$	Para cada $t \in (0, T_{MAX})$
$g(x) = x(1-x)$	Para cada $x \in (0,l)$

(i) Solução Exata

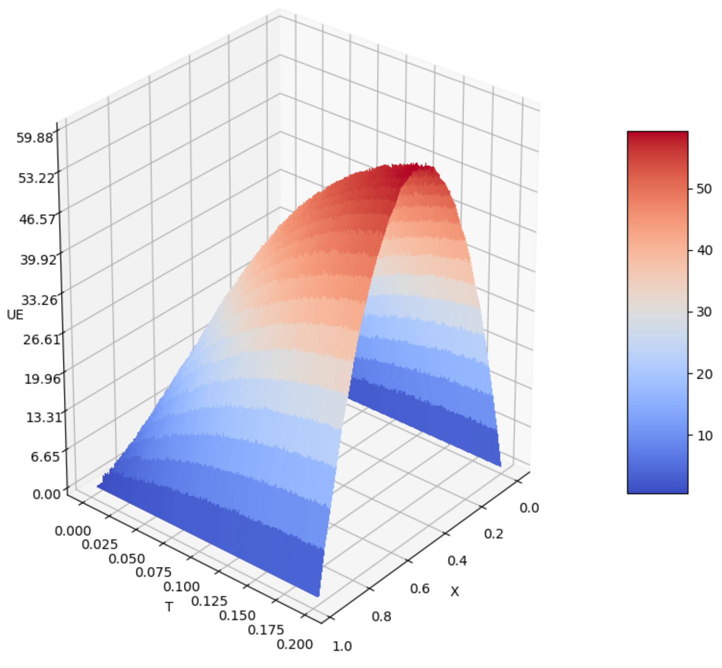
A solução exata para os parâmetros fixados em $k = (0.5, 0.5)$ foi encontrada com o método das diferenças finitas e o gráfico obtido é o exibido abaixo:

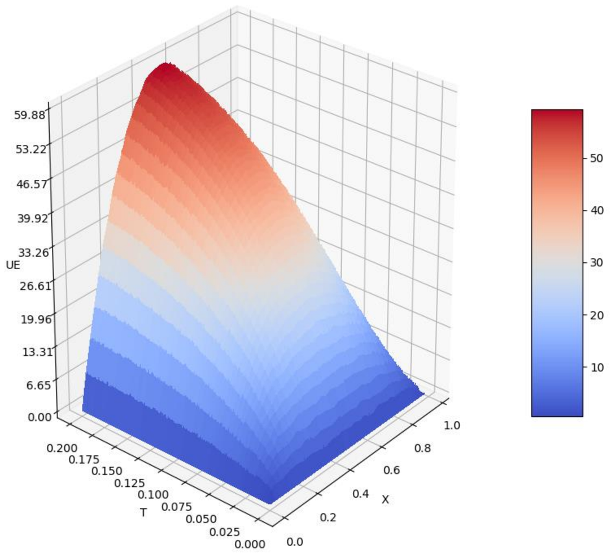




(ii) Solução com Ruído

O Esboço da função com o “ruído” é apresentada abaixo:





(iii) Integração Numérica

Foi implementado uma rotina em python de diferenças finitas, o código completo está em anexo:

L	TMax	Total Pontos X	Total Pontos T	DX	DT	S
1	0.2	35	500	0.0285	0.0004	0.49

```
def solucaoNumerica(U, _K1, _K2):
    #Diferenças Finitas
    for j in range (0, numpontosx):
        U[0,j] = ((j*DX)*(1-(j*DX)))

    U[0,numpontosx-1] = 0
    #print(U[0,:])

    # iteracao
    for n in range (1, numpontost):
        U[n,0] = 0
        U[n,numpontosx-1] = 0
        for j in range (1, numpontosx -1):
            U[n,j] = S * (U[n-1,j+1] - 2*U[n-1,j] + U[n-1,j-1]
                          + _K1*np.exp(-_K2*j*DT)) + U[n-1,j]
            #print(U[n,:])
    return U
```

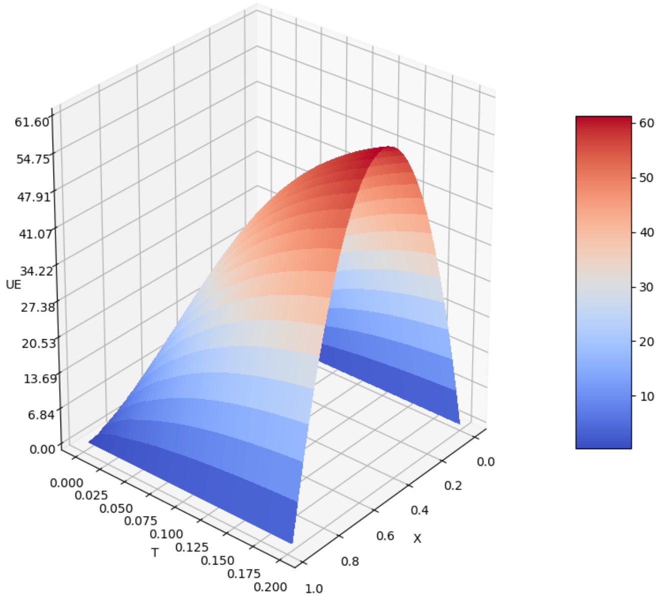
(iv) Estimação do Paramêtro k

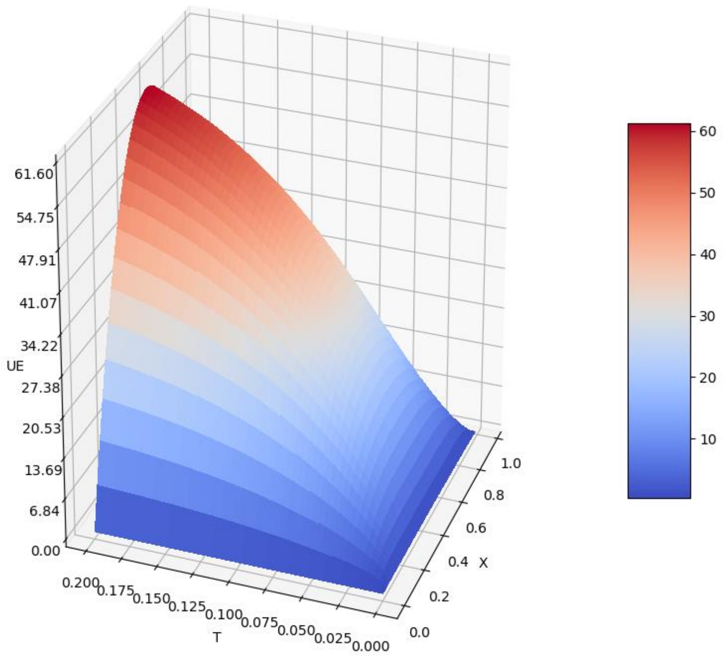
Determinar os valores de k que aproximam a função com ruído que simula a função real. Foram executadas 100 simulações com valores de K crescentes e 100 simulações com K aleatório.

(a) Método 1

Parâmetros obtidos

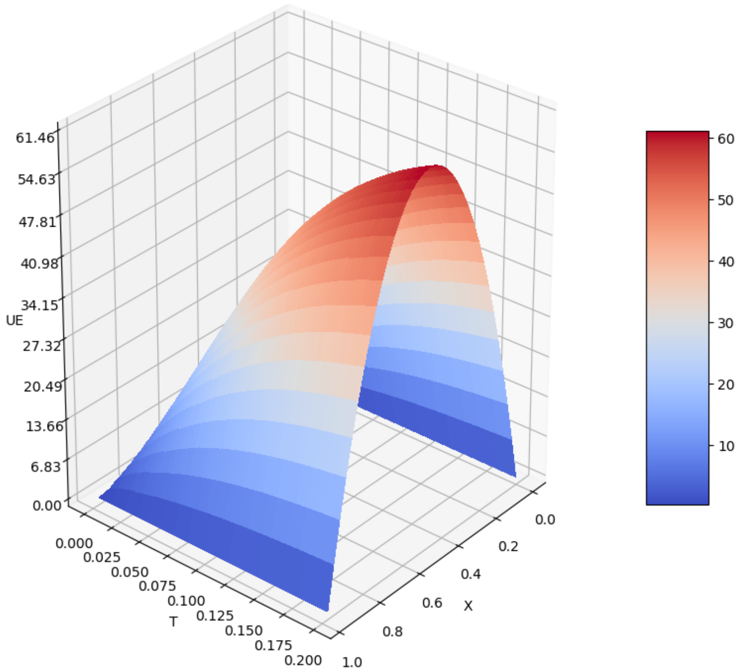
Mínimo J(k)	K1	K2
8987.776521541713	0.49	0.49

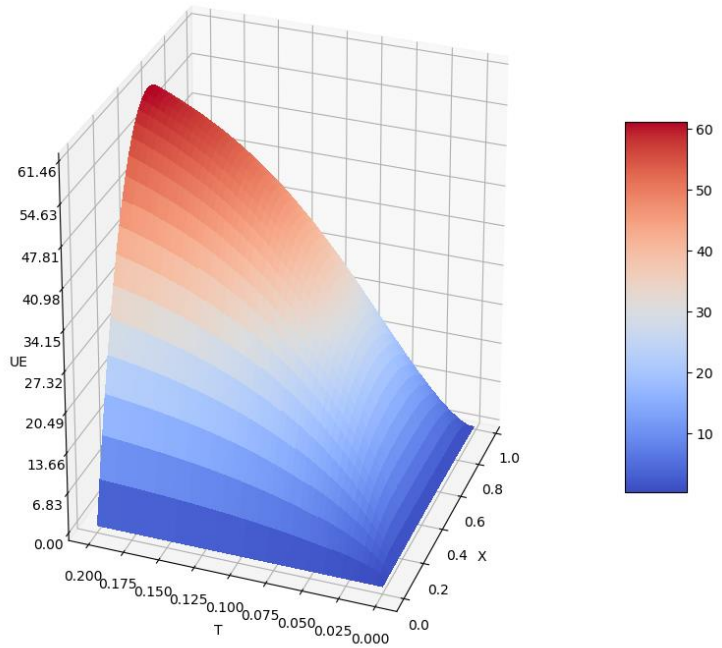




(b) Método 2

Mínimo J(k)	K1	K2
8662.025696588042	0.490058543000771	0.827694316260286

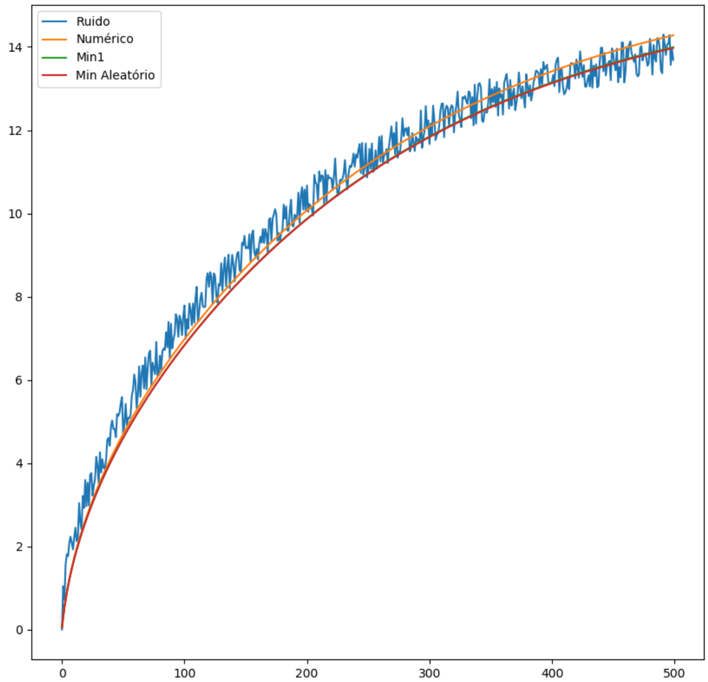




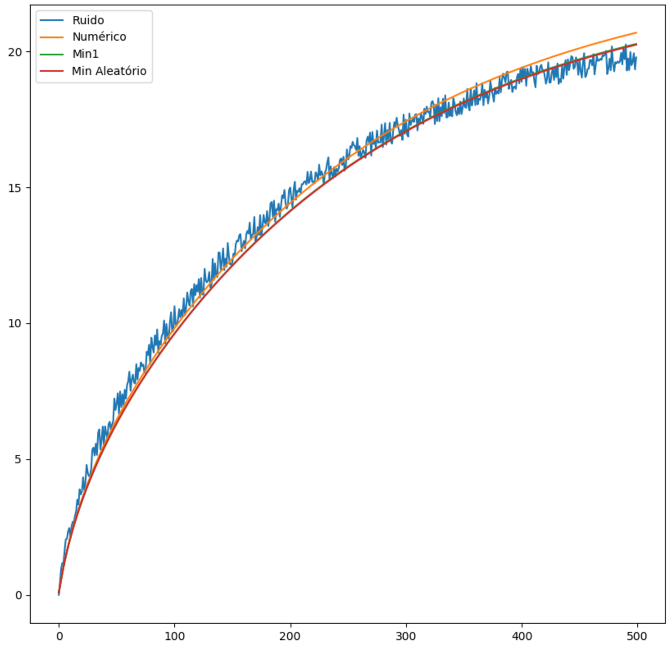
(v) Representação Gráfica

Representação gráfica do perfil do eixo X dos métodos calculados para comparar as distâncias obtidas:

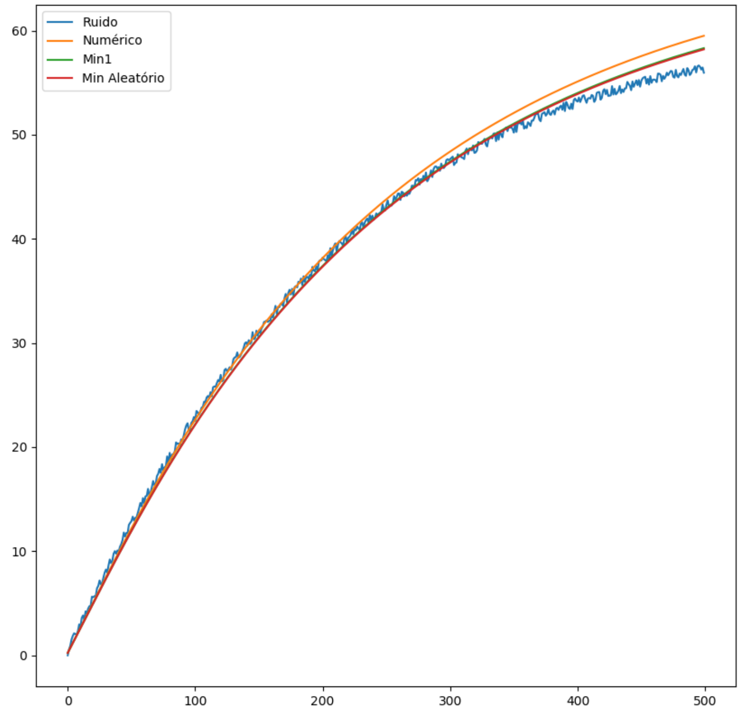
X = 0.057



X = 0.0855

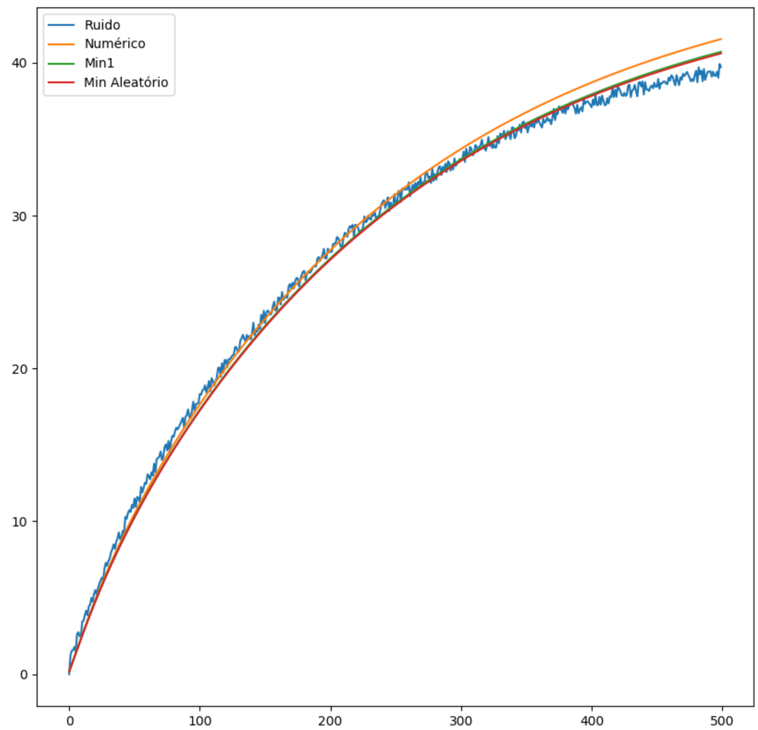


X = 0.37



Ruido
Numérico
Min1
Min Aleatório
60
50
40
30
20
10
0
0
100
200
300
400
500

X = 0.7695



Ruido
Numérico
Min1
Min Aleatório
40
30
20
10
0
0
100
200
300
400
500

Conclusão

Ao rodar a simulação variando os valores de k1 e k2 percebeu-se que para $k1 < 0$ a temperatura do sistema vai diminuir e para valores de $k1 > 0$ a temperatura do sistema aumenta. Com relação a k2, para $k2 > 0$ o sistema é estável e tende para a função de forma suave no infinito, porém para $k2 < 0$ o sistema torna-se instável.

Isso comprova-se analisando a distância do gráfico da modelagem obtida com o melhor k1 e k2 obtido através da minimização, e o gráfico da função com ruído. Nota-se que os gráficos ficam mais próximos e suaves, ou seja, ao avançar no espaço e no tempo, a função calculada torna-se mais próxima e suave em relação ao ruído.

Mesmo a solução obtida com $k1 = 0.5$ e $k2 = 0.5$ o erro não é muito grande e poderia ser usada dependendo da aplicação.