Escola Politécnica da USP
Departamento de Matemática Aplicada

MAP3121 – MÉTODOS NUMÉRICOS E APLICAÇÕES

EXERCÍCIO PROGRAMA II

UM PROBLEMA INVERSO PARA A OBTENÇÃO DE DISTRIBUIÇÃO DA TEMPERATURA (EQUAÇÃO DO CALOR)

Andre Devay Torres Gomes – N° USP: 10770089

Filipe Coutinho Rodrigues – N° USP: 10769751

Turmas 01 e 07 – Professores Marcone Pereira e Saulo Rabello

São Paulo

Junho de 2020

SUMÁRIO

Introdução **2**

Objetivos **2**

Análise Geral do Problema **2**

O Problema Físico 2

Modelo Computacional 3

Equação 3

Equações Diferenciais Parciais (EDP) 4

Problema Transiente 4

Equação Parabólica 5

Condições iniciais e condições de fronteiras 5

Princípio da Superposição 5

Discretização 6

Malha 6

Definição do método dos mínimos quadráticos 6

Abordagem a partir do menor erro 7

Sistema Normal 8

Análise das Tarefas **8**

Tarefa A 9

Tarefa B 9

Tarefa C 10

Testes Realizados **16**

Item A 16

Item B 17

Item C 18

Item D 23

Conclusão **27**

Referências **27**

1. Introdução

Dentro da natureza, podemos definir as análises de seus fenômenos em dois tipos, os *Problemas diretos* e os *Problemas inversos.* Enquanto o primeiro tenta determinar o efeito de um fenômeno a partir de uma causa, o segundo procura definir a causa a partir de um determinado efeito natural.

A conceituação do calor e seu fenômeno de distribuição é objeto de estudo da humanidade há muitos séculos devido a sua importância e influência em quase todos os acontecimentos na natureza. Desde a Idade da Pedra, a distribuição de calor acompanha a evolução dos seres humanos e é fundamental para o progresso da humanidade.

Gradativamente, as ferramentas para estudos e observação dos fenômenos naturais vêm evoluindo e, com o avanço da tecnologia, uma nova alternativa de análise se tornou viável: o modelo computacional. Assim, conseguimos trazer soluções numéricas de alta complexidade e com inúmeras etapas de forma mais ágil e eficaz, pois transferimos o trabalho matemático para uma máquina.

2. Objetivos

O EP1 foi enunciado como um *Problema Direto* e nele tivemos que determinar a evolução e comportamento de uma temperatura ao longo de uma barra sujeita a fontes de calor, a partir de uma dada distribuição.

Já o EP2, traz uma continuação do problema proposto de distribuição de temperatura. Entretanto, desta vez, foi enunciado um *Problema Inverso.* Ou seja, a partir do conhecimento da distribuição de temperatura ao longo de uma barra (efeito), devemos encontrar, analisar e discutir as causas dessa problemática.

3. Análise Geral do Problema

3.1. O Problema Físico

No nosso problema, temos uma barra fictícia (de largura e altura inexistentes e/ou desprezíveis), tornando o nosso problema *unidimensional.* Podemos perceber, através da equação (1), que a barra não é termicamente isolada, pois ela tem influência direta de uma fonte externa no seu comportamento de distribuição de temperatura. Além da influência dessa fonte externa, é perceptível que existe uma influência devido a temperatura inicial da barra e do comportamento de seus extremos,

respectivamente representados pelas equações (2), (3) e (4) do enunciado. Para o EP2 especificamente, consideramos as condições iniciais e fronteiras como nulas (ou seja, u0(x) = g1(t) = g2(t) = 0) .

Em determinadas questões do EP1 e todas as questões do EP2 também é adicionado a conceituação da discretização da força f. Para cada ponto p de nossa barra, haverá uma intensidade e limite de força associado (g_h), cada limite de força associado terá influência sobre um intervalo h deste ponto p, ou seja, entre [p-h/2 ; p+h/2] e, assim, a função f(t,x) da força será composta pelo produto da intensidade (regida temporalmente) e do limite de força (regido espacialmente).

3.2. Modelo Computacional

A Modelagem computacional consiste em uma área de conhecimento multidisciplinar que trata da aplicação de modelos matemáticos e técnicas da computação à análise, compreensão e estudo da fenomenologia de problemas complexos em outras áreas do conhecimento. Aqui, como já dito anteriormente, iremos tratar de um problema físico de distribuição de temperatura.

Para o desenvolvimento de qualquer modelo computacional, devemos separar nossa análise e estudo em dois passos principais. Em um primeiro momento, necessitamos compreender e analisar quais são as grandezas físicas que estão envolvidas no fenômeno e como elas se correlacionam. No segundo momento, aplicamos os conceitos físicos já conhecidos por nós.

Para tornarmos o modelo computacional o mais próximo e condizente a realidade, devemos nos preocupar com dispor de equações que representam bem o fenômenos, definir as regiões nas quais estas equações são válidas e analisar as limitações existentes para cada problema específico. Dessa forma somos capazes de avaliar as opções de resolução de uma maneira muito mais prática, executando o modelo de maneira mais limpa e eficaz, o que simplifica a obtenção de resultados satisfatórios para o problema que buscamos solucionar.

Dessa forma, podemos dizer que a modelagem computacional consiste em um meio com potencial altamente eficaz de se solucionar diversos problemas de maneira ágil e concreta. Por isso que, atualmente, esse método de resolução é amplamente utilizado por sua praticidade e agilidade, que compensam e otimizam processos por meio de seus algoritmos pré-programados.

3.3. Equação

Tanto a parte I, quanto a parte II dos nossos exercícios programas se baseiam na Equação de Difusão de Calor (1). Desta maneira, faremos algumas considerações sobre ela.

$$u_t(t,x) = u_{xx}(t,x) + f(t,x)$$

3.3.1. Equações Diferenciais Parciais (EDP)

De forma genérica e vaga, podemos tentar conceituar uma equação diferencial, como uma equação ao que estabelece relações entre uma função ou um conjunto de funções e as suas derivadas (até uma certa ordem). Para equações diferenciais ordinárias, a dimensão do domínio deve ser igual a 1. Já para as EDPs, devemos ter a dimensão do domínio de u necessariamente maior que 1.

$$F(x, y(x), y'(x), y''(x), ..., y^{(n)}(x)) = 0$$

Fórmula generalizada para as EDOs

$$F(x_1, ..., x_n, u, \frac{\partial u}{\partial x_1}, ..., \frac{\partial u}{\partial x_n}, \frac{\partial^2 u}{\partial {x_1}^2}, ..., \frac{\partial^2 u}{\partial x_1 \partial x_n}, \ ..., \frac{\partial^k u}{\partial {x_n}^k}) = 0$$

Fórmula generalizada para as EDPs

Percebe-se que, neste caso, temos as variáveis independentes x (uma dimensão física) e t (uma dimensão temporal), ou seja, há duas dimensões para o nosso problema de distribuição de temperatura.

Uma grande diferença entre EDOs e EDPs é a informação suplementar necessária para a unicidade de solução. Na solução de uma EDO linear, por exemplo, tem-se uma ou mais constantes arbitrárias: podemos determinar estas constantes impondo condições iniciais, isto é, fixando os valores da solução e de suas derivadas até certa ordem em um determinado ponto; podemos também obter unicidade no caso de intervalos finitos, impondo condições nos extremos dos intervalos, as chamadas condições de contorno. Para uma EDP mesmo que linear, contudo, a solução geral envolve funções arbitrárias das variáveis independentes, de modo que existe um grau de generalidade muito maior com relação à forma da solução.

Por fim, vale ressaltar a relevância social das equações diferenciais parciais, pois encontramos estas sendo ferramenta para o estudo da ondulatória, eletrostática, mecânica dos fluidos, entre outras diversas áreas do conhecimento.

3.3.2. Problema Transiente

Dentro das classificações utilizadas para as situações problemas, podemos definir um grupo de equações que tem como variável independente o tempo (os problemas transientes) e um outro grupo que independe da variação temporal (os problemas estacionários). Fisicamente, esta conceituação e classificação é importante, pois, na maior parte das vezes, o estudo de fenômenos estacionários é possível (ou um pouco mais conclusivo) de ser feito de forma empírica, do que um problema transiente.

Em problemas determinados pela variação temporal, devido a percepção humana do tempo ser limitada, os problemas são menos conclusivos e a dependência de ferramentas auxiliares (como a

modelagem computacional) é muito maior. Por fim, podemos perceber que a problemática apresentada no enunciado é um *problema transiente.*

3.3.3. Equação Parabólica

Equações parabólicas consistem em um tipo de equações diferenciais parciais de segunda ordem, das quais podemos obter soluções a partir de determinado métodos numéricos, entre eles alguns presentes no conteúdo desta matéria.

Quando aplicamos estas equações em uma equação de calor, podemos perceber que a variável do tempo é fortemente considerada, e podemos notar também que este tipo de sistema pode ser transformado em uma matriz (a partir de seus coeficientes) que deverá apresentar um determinante igual a zero.

3.3.4. Condições iniciais e condições de fronteiras

Nas problemáticas das equações diferenciais, além da equação que rege o problema (propriamente dito), é necessário a especificação de condições suplementares para ser viável uma solução única do problema. Para todas as EDPs que podem ser classificadas como equações de evolução, as soluções são unicamente determinadas pelas condições iniciais impostas.

Assim, podemos definir nossa situação do *problema direto* como um *problema de Cauchy*, pois precisamos estudar as soluções de uma equação de evolução (a u(t,x)) em função das condições iniciais dadas, representada pela nossa equação (2).

Ademais, há condições de fronteiras explicitadas. Essas são uma das classes de condições suplementares amplamente utilizada nas equações diferenciais. Quando consideramos um domínio Ω com fronteira $\partial\Omega$, as condições de fronteira prescreverão o comportamento das soluções procuradas sobre a "parte espacial" de $\partial\Omega$ (que é a região limítrofe do domínio Ω).

Dentro do nosso enunciado, podemos perceber que as equações (3) e (4), seguem a ideia das condições de fronteira do tipo *Dirichlet*, sendo prescritos o comportamento que ocorre nos extremos da barra estudada por nós.

3.3.5. Princípio da Superposição

Uma outra propriedade fundamental desta classe de equações diferenciais e utilizada na resolução da parte II de nosso exercício programa é o Princípio da Superposição. Baseada na aplicação do Método de Separação de Variáveis de EDPs, podemos definir L como um operador diferencial parcial linear de ordem k cujos coeficientes estão definidos em um aberto $\cap \subseteq R^n$. Suponha que $\{u_m\}$ é um conjunto de funções de classe C^k em $\cap$ satisfazendo a EDP linear homogênea L*u = 0. Então, se $\{a_m\}$ é uma sequência de escalares tal que a série:

$$u(x) = \sum_{m=1}^{\infty} \alpha_m u_m(x)$$

é convergente e k vezes diferenciável termo a termo em $\cap$, então u satisfaz Lu = 0.

É partindo desta ideia, portanto, que conseguimos dividir o nosso $u_T(x)$ na série descrita pela equação (38) no enunciado. E, através do método dos mínimos quadrados, encontrar as parcelas $\{a_m\}$ da problemática (perceba que esses coeficientes são enunciados no EP2 como a_k).

3.4. Discretização

Dentro da matemática, a discretização é o processo de transferência de funções contínuas, modelos, variáveis e equações em contrapartes discretas. A ideia é dividirmos funções complexas e dificilmente entendíveis por um computador em inúmeras expressões aritméticas mais simples (que para um humano demoraria muito tempo para fazer, mas para um computador é muito mais simples por haver mais contas e menor complexidade de raciocínio). De uma forma simplificada, podemos entender o processo de discretização como responsável por transcrever as Equações Diferenciais Parciais (EDPs) em expressões que conseguem ser interpretadas pelo computador.

No caso do nosso problema enunciado, as principais discretizações são a temporal (analisamos o tempo em pequenas parcelas discretas) e na dimensão espacial 1D (ao longo do comprimento da barra).

3.5. Malha

Quando falamos em modelos computacionais, que temos métodos iterativos como meio de se solucionar os problemas propostos, podemos utilizar discretizações como forma de obtermos aproximações numéricas mais precisas. Ao realizarmos diferenças finitas, introduzimos uma malha espacial que coordena as aproximações nas duas variáveis disponíveis: tempo e espaço.

Essa malha é dada no nosso problema proposto pela composição dos pontos $x_i = i\Delta x$ (com i = 0,1…N), com $\Delta x = 1/N$ no espaço. Para a discretização temporal utilizamos os pontos $t_k = k\Delta t$ (com k = 0,1...M) e $\Delta t = T/M$, sendo todos estes pontos obtidos salvos na malha final localizada em $u(t_k, x_i)$.

3.6. Definição do método dos mínimos quadráticos

O método dos mínimos quadrados se apoia sobre a ideia de uma aproximação dos valores buscados por meio de uma extrapolação, com uma certa confiança, dos dados do método. Dessa forma, somos capazes de obter melhores resultados que se adequem às soluções buscadas, com redução do seu erro final.

Para isso, estabelecemos uma combinação linear das funções com as quais estamos trabalhando, para que possamos, ao final, encontrar uma função aproximadora que detenha os melhores valores passíveis de se obter. Esta função que buscamos obter apresenta um caráter linear, já que sua ideia seria, especificamente, aproximar os polinômios.

3.6.1. Abordagem a partir do menor erro

Para encontrarmos um erro mínimo dos resultados que visamos neste EP2, utilizamos o método dos mínimos quadráticos. A partir do conhecimento das funções $u_k(t, x)$ e suas forçantes, podemos determinar valores de intensidade presentes em cada uma delas, a_k, para minimizarmos o valor do erro:

$$E = [\Delta x \sum^{N-1}(u_T(x_i) - \sum^{nf} a_k u_k(T,x_i))^2]^{½}$$

Retiramos os valores da equação de uma forma que, como resultado, obteremos um erro que seja o menor possível, e os aplicamos à fórmula ou equação, para que o valor adequado para o modelo computacional seja obtido e possa ser utilizado.

Dessa forma, ao maximizarmos a parcela final do somatório das intensidades $[a_k u_k(T,x_i))]$, podemos tornar seu valor mais próximo ao do $u_T(x_i)$ o que irá reduzir o somatório na equação e, consequentemente, tender o erro a zero.

Para que essa minimização do erro proposto ocorra, devemos derivá-lo em relação aos valores dispostos das funções u_k e igualar estas operações a zero. Dessa forma somos capazes de iniciar o procedimento que nos fará atingir nosso objetivo, como pode ser visto logo abaixo:

$$\frac{\partial E}{\partial a_i} = 0$$

Se tomarmos uma derivação genérica como referência, podemos obter por fim um somatório de equações que subtraem da função principal um agregado de valores referentes à multiplicação das funções u com as intensidades:

$$\frac{\partial E}{\partial a_i}(a_1, a_2, ..., a_m) = \frac{\partial \sum}{\partial a_i}(u_T(x_i) - (a_1.u_{1k}(T, x_i) + ... + a_m.u_{nfk}(T, x_i)))^2$$

Ao trabalharmos com essa equação, somos capazes de separá-la, após desenvolver seus equacionamentos. Após a destrincharmos, percebemos que as multiplicações presentes consistem em

produtos internos das funções, com seus respectivos coeficientes:

$$a_1 . < u_{1k}, u_{1k} > + a_2 . < u_{2k}, u_{1k} > + ... + a_m . < u_{nfk}, u_{1k} > = < u_{1k}, u_T >$$

Ao realizarmos estas operações para todos os possíveis valores trabalhados, nós alcançamos, por fim, um sistema normal. Como mostrado na figura abaixo:

$$\begin{bmatrix} < u_1, u_1 > & < u_2, u_1 > & \cdots & < u_{nf}, u_1 > \\ < u_1, u_2 > & < u_2, u_2 > & \cdots & < u_{nf}, u_2 > \\ \vdots & \vdots & \vdots & \vdots \\ < u_1, u_{nf} > & < u_2, u_{nf} > & \cdots & < u_{nf}, u_{nf} > \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{nf} \end{bmatrix} = \begin{bmatrix} < u_T, u_1 > \\ < u_T, u_2 > \\ \vdots \\ < u_T, u_{nf} > \end{bmatrix}$$

Este sistema final, obtido através dos produtos internos e funções ou números, se alinha na forma de matrizes e apresenta solução única, que minimiza a função central do problema, aproximando-a ao máximo do valor desejado.

Vale ressaltar que o produto interno definido para o espaço vetorial que estamos utilizando no exercício programa II é feito da seguinte forma:

$$< u, v > = \sum_{i=1}^{N-1} u(x_i) v(x_i)$$

3.6.2. Sistema Normal

Quando temos um sistema linear com vetores linearmente independentes, dizemos que a matriz desse sistema possui posto completo e, portanto, o sistema admitirá apenas solução única. Como os elementos presentes nessa matriz são resultados de multiplicações entre as funções que estamos trabalhando, podemos concluir que cada elemento da matriz seja um produto interno das funções, enquanto os elementos do lado direito correspondem ao produto interno do vetor das medidas calculadas com uma particular coluna destas mesmas funções aproximadoras.

Os vetores do produto interno deste tipo de sistema por vezes podem ser ortogonais, o que nos permite concluir, e consequentemente simplificar os modelos, que seus produtos internos resultam em zero quando i ≠ j, o que torna o sistema diagonal.

4. Análise das Tarefas

4.1. Tarefa A

Utilizando como base as equações de distribuição de temperatura discutidas amplamente durante o EP1, somos capazes de gerar os vetores parciais da solução (os uk(T, xi)) que serão usados, por sua vez, ao longo desta segunda parte do problema inverso. Utilizando as condições de contorno pré-estabelecidas, podemos achá-los através da integração das equações de calor, por meio do método de Crank-Nicolson já desenvolvido. Diferentemente do EP1, não necessitamos dos uk's em toda a discretização do tempo (somente em t = T), portanto, modificamos tal função para que só seja retornado a parcela dos uk's desejados no EP2.

Outra pequena modificação realizada nesta função foi em relação ao cálculo das forçantes. Ajustamos a função da intensidade que é a mesma durante toda a realização do EP2 e trouxemos a variável das fontes pontuais p para dentro do método de Crank-Nicolson (para que consigamos conceituar a força para cada nova fonte pontual p).

Vale ressaltar que com o M = N em todo o problema descrito para o exercício programa II, temos que:

$$\lambda = \frac{\Delta t}{\Delta x^2} \quad ; \quad \Delta t = \frac{1}{M} \quad \text{(pois T = 1)} \quad ; \quad \Delta x = \frac{1}{N}$$

$$\lambda = \frac{\frac{1}{M}}{\frac{1}{N^2}} \quad \Rightarrow \quad \lambda = \frac{1}{M}.N^2 \quad ; \quad \text{se} \quad M = N \quad \Rightarrow \quad \lambda = N$$

Ou seja, o nosso valor de ƛ será igual a N.

4.2. Tarefa B

Assim como foi definido na seção do "sistema normal", nesta tarefa tivemos de montar uma matriz normal do problema de mínimos quadrados para podermos calcular as intensidades a partir da minimização do erro. Para isso, definimos o sistema por meio das matrizes que o compõem: a matriz dos produtos internos feitos entre as medições obtidas com o programa, a matriz coluna das intensidades (valores buscados nos testes) e a matriz resultante dos produtos internos entre os vetores e colunas.

$$\begin{bmatrix} < u_1, u_2 > & . & . & . & < u_{nf}, u_1 > \\ . & & . & & . \\ . & & . & & . \\ . & & . & & . \\ < u_1, u_{nf} > & . & . & . & < u_{nf}, u_{nf} > \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ . \\ . \\ . \\ a_n \end{bmatrix} = \begin{bmatrix} < u_T, u_1 > \\ . \\ . \\ . \\ < u_T, u_{nf} > \end{bmatrix}$$

Deste sistema, fomos capazes de achar o vetor solução unitário, já que o sistema aceita somente uma única solução, que aproxima os valores das intensidade ao máximo daqueles que desejamos.

Assim, podemos definir um pseudocódigo de quais passos foram necessários realizar para montar o sistema normal definido no enunciado do EP2:

1) Dados os pontos p1,... , pnf, gerar os vetores uk através do método de Crank-Nicolson modificado.

2) Calcular os produtos internos necessários para cada elemento

3) Montar as matrizes A e b (seguindo o formato A*x = b)

E, a partir da definição do nosso pseudocódigo, fomos capazes de trabalhar com essa matriz no código, ao criarmos uma função própria para realizar a operação desta na qual desenvolvemos o seguinte método:

```
def sistema_normal(M, N, nf, uT):

  # Calculando cada uk para um pnf especifico
  vetorNF = []
  for k in range (len(nf)):
    vetorNF.append(tarefa_2c_adapt(N, M, nf[k]))


  # Criando a matriz dos produtos internos <uk, uk> para cada k de 1 a nf (chamado de a)
  # Criando o vetor dos produtos internos <uT, uk> para cada k de 1 a nf (chamado de b)
  A = []
  B = []
  for i in range (len(nf)):
    m = []
    B.append(np.dot(uT, vetorNF[i]))
    for j in range (len(nf)):
        a = np.dot(vetorNF[i], vetorNF[j])
        m.append(a)
    A.append(m)
```

4.3. Tarefa C

A fatoração $A = LDL^t$ consiste em fatorar a matriz A, que é n x n, em uma multiplicação de três matrizes, onde L é uma matriz triangular inferior e D é uma matriz diagonal com números positivos (perceba que L^t é a transposta da matriz L). Note que, nesse caso, diferentemente da decomposição na parte I do exercício programa, temos uma matriz-não esparsa, ou seja, a matriz A

não possui uma grande quantidade de elementos com valor zero. Outra característica importante ao nosso problema é que a matriz A é simétrica, assim, seus componentes $Aij = Aji$.

Dado tal matriz A de n x n, para elemento do tipo Aij, devemos definir um componente Vj, que será igual ao Lij * Dj (onde as componentes Lij e Dj pertencem as matrizes L e D).

$$\begin{bmatrix} 1 & 0 & 0 \\ l21 & 1 & 0 \\ l31 & l32 & 1 \end{bmatrix} . \begin{bmatrix} d1 & 0 & 0 \\ 0 & d2 & 0 \\ 0 & 0 & d3 \end{bmatrix} = \begin{bmatrix} d1 & 0 & 0 \\ d1.l21 & d2 & 0 \\ d1.l31 & d2.l32 & d3 \end{bmatrix}$$

Ademais, por D ser uma matriz diagonal, não faz sentido se falar em Dij, pois para qualquer elemento não nulo dessa matriz, i deve ser igual a j.

Em seguida, definimos o algoritmo a ser utilizado na fatoração, que pode ser facilmente modificado graças ao fato da matriz ser simétrica. Dessa forma, podemos descrever o processo que aplicamos na matriz de maneira mais simplificada, o que nos auxilia na formatação do próprio programa a ser executado pelo computador.

Para exemplificar os passos, podemos utilizar como exemplo uma matriz A genérica que tenha seu tamanho 3 x 3:

$$A = \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l21 & 1 & 0 \\ l31 & l32 & 1 \end{bmatrix} . \begin{bmatrix} d1 & 0 & 0 \\ 0 & d2 & 0 \\ 0 & 0 & d3 \end{bmatrix} . \begin{bmatrix} 1 & l12 & l13 \\ 0 & 1 & l23 \\ 0 & 0 & 1 \end{bmatrix}$$

O que resulta em:

$$\begin{bmatrix} d1 & d1.l21 & d1.l31 \\ d1.l21 & d2 + d1.l21^2 & d2.l32 + d1.l21.l31 \\ d1.l31 & d1.l21.l31 + d2.l32 & d1.l31^2 + d2.l32^2 + d3 \end{bmatrix}$$

Em seguida, podemos definir um sistema linear associando os componentes de A com os componentes das outras matrizes (L e D), chegando em:

$$\begin{cases} a11 = d1 \\ a31 = a13 = d1.l31 \\ a32 = d1.l21.l31 + d2.l32 \end{cases} \qquad \begin{cases} a21 = a12 = d1.l21 \\ a22 = d2 + d1.l21^2 \\ a33 = d1.l31^2 + d2.l32^2 + d3 \end{cases}$$

Podemos também equacionar a mesma ideia em uma matriz genérica A que seja 4 x 4:

$$A = \begin{bmatrix} d1 & d1.l21 & d1.l31 & d1.l41 \\ d1.l21 & d2 + d1.l21^2 & d2.l32 + d1.l21.l31 & d2.l42 + d1.l21.l41 \\ d1.l31 & d1.l21.l31 + d2.l32 & d1.l31^2 + d2.l32^2 + d3 & d1.l31.l41 + d2.l32.l42 + d3.l43 \\ d1.l41 & d1.l41.l21 + d2.l42 & d3.l43 + d1.l14.l31 + d2.l32.l24 & d4 + d1.l41^2 + d2.l42^2 + d3.l43^2 \end{bmatrix}$$

E, assim , conseguimos perceber, que existe um padrão entre a construção das equações lineares e podemos definir nosso um pseudocódigo no seguinte formato:

1. Entrar (“INPUT”) com a dimensão n e as entradas a_{ij} da matriz A, para $1 \leq i,j \leq n$ (“for”).
2. Encontrar os valores das entradas das matrizes L, l_{ij} para $1 \leq j \leq i$ e $1 \leq i \leq n$, e D, para d_i $1 \leq i \leq n$.
3. Definir o range do código de acordo com os limites das matrizes (sendo i = 1,... n e j = 1,... i-1) e escrever as equações que usaremos para obter as entradas desejadas (que estão escritas logo abaixo deste pseudocódigo).
4. Compilar os valores encontrados de l_{ij} e d_i, chamando-os ao fim do código para que possam ser utilizados fora da função (“OUTPUT”).

A partir do pseudocódigo, conseguimos definir as fórmulas descritas abaixo que relacionam de maneira matemática e simplificada os procedimentos que geram o formato final das matrizes em questão (D e L), presentes na decomposição realizada.

$$D_j = A_{jj} - \Sigma L_{ij}.D_i$$

$$L_{ij} = 1/D_j(A_{ij} - \Sigma L_{kj}.L_{ji}.D_i) \text{ , se } i > j$$

Por fim, transcrevemos o nosso código para o formato de Python como está mostrado na função def LDL_decomp(A):

```
def LDL_decomp (A):
  D = []
  L = []
  for i in range (tamanho):
      for j in range (tamanho):
          d  = [0] * tamanho
          l = [0] * tamanho
      D.append(d)
      L.append(l)

  #Passo 1
  for i in range (tamanho):
      soma0 = 0
      v = []

      #Passo 2
```

```
        for j in range (i):
            e = float(L[i][j]) * float (D[j][j])
            v.append(e)
            soma0 = soma0 + float(L[i][j]) * float(L[i][j]) * float (D[j][j])

        #Passo 3
        D[i][i] = A[i][i] - soma0

        #Passo 4
        for j in range (i, tamanho):
            soma1 = 0
            for k in range (i):
                soma1 = soma1 + float(L[j][k]) * float (v[k])
            L[j][i] = ((A[j][i] - soma1)/(float(D[i][i])))

    #Retornando:
    matrizes3 = [L, D]
    return matrizes3
```

Depois desta etapa, nos voltamos para a manipulação do sistema linear, a fim de encontrarmos o vetor que contém a solução das intensidades, o que nos levou a aplicar um algoritmo similar ao utilizado no EP1 para a solução do problema, e que será desenvolvido a seguir:

Utilizando como exemplo inicial um modelo de matrizes; $3 \times 3 \rightarrow L.D.L^t.x = b$

$sendo: D.L^t.x = z;$ Obtemos a equação:

$$\begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{23} & 1 \end{bmatrix} . \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

Ao desenvolvermos as operações matriciais, originamos o sistema:

$$\begin{cases} z_1 = B_1 \\ z_1.l_{21} + z_2 = B_2 \\ z_1.l_{31} + z_2.l_{23} + z_3 = B_3 \end{cases}$$

Isolamos, para conseguirmos encontrar os valores de z. Em seguida, já com o vetor z definido, podemos aplicar a mesma ideia considerando o $L^t x = y$. Montamos um novo produto de matrizes:

$$D.y = z$$

Da qual operamos da mesma forma realizada anteriormente:

$$\begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix} . \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}$$

E em seguida transformamos, também, em um sistema linear:

$$\begin{cases} d_1 . y_1 = z_1 \\ d_2 . y_2 = z_2 \\ d_3 . y_3 = z_3 \end{cases}$$

Assim, conseguimos encontrar os valores do vetor y e podemos definir nossa terceira e última equação matricial:

$$L^t . x = y$$

$$\begin{bmatrix} 1 & l_{12} & l_{13} \\ 0 & 1 & l_{23} \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Da qual obtemos o último sistema desta sequência:

$$\begin{cases} x_1 + x_2 l_{12} + x_3 l_{13} = y_1 \\ x_2 + x_3 l_{23} = y_2 \\ x_3 = y_3 \end{cases}$$

Com isso encontramos o valor de x e, após uma análise, somos capazes de encontrar um padrão e desenvolver os 3 sistemas lineares para matrizes de tamanho n:

$$\begin{bmatrix} 1 & 0 & . & . & . & 0 \\ l_{12} & 1 & 0 & . & . & 0 \\ . & & . & & & \\ . & & & . & & \\ . & & & & . & \\ l_{1,n} & . & . & l_{n-1,n} & & 1 \end{bmatrix} . \begin{bmatrix} z_1 \\ . \\ . \\ . \\ . \\ z_n \end{bmatrix} = \begin{bmatrix} B_1 \\ . \\ . \\ . \\ . \\ B_n \end{bmatrix}$$ (equação matricial de ordem n x n)

Assim, desenvolvemos o sistema linear e, através dele, já somos capazes de perceber o padrão que está inicialmente implícito no algoritmo para a matriz 3 x 3:

$$\begin{cases} z_1 = B_1 \\ z.l_{21} + z_2 l_{32} + z_3 = B_4 \\ . \\ . \\ . \\ z_1.l_{n,1} + z_2.l_{n,2} + ... + z_{n-1}.l_{n,n-1} + z_n = B_n \end{cases}$$

Por conseguinte, vemos o padrão no sistema desenvolvido a partir da segunda e terceira equações matriciais previamente mostradas, que serão da forma:

$$\begin{cases} y_1 = \frac{z_1}{d_1} \\ y_2 = \frac{z_2}{d_2} \\ . \\ . \\ . \\ y_n = \frac{z_n}{d_n} \end{cases}$$

$$\begin{cases} x_1 + x_2.l_{n,2} + ... + x_n.l_{n,2} = y_1 \\ . \\ . \\ . \\ x_{n-1} + x_n.ln, n-1 = y_{n-1} \\ x_n = y_n \end{cases}$$

Podemos, a partir disso, descrever um algoritmo de resolução para este tipo específico de decomposição de matrizes na linguagem Python. No exercício programa, desenvolvemos este algoritmo da seguinte forma:

```
    # Encontrando x em LDLt*x = B
    Adecomposta = LDL_decomp(A)

    # Manipulação de sistema linear similar a ideia do EP1 (porém com algumas modificações)
    # Ax=b -> LZ=b + DY=Z + L_tx=Y

    L = Adecomposta[0]
    D = Adecomposta[1]

    Z=[]
    Z.append(B[0])
    for i in range (1,len(B)):
        soma0 = 0
        for j in range (i):
            soma0 = soma0 + float(Z[j]) * float (L[i][j])

        z_novo = B[i] - soma0
        Z.append(z_novo)


    Y=[]
    for j in range (len(Z)):
        y=float(Z[j])/float(D[j][j])
        Y.append(y)

    X=[]
    for a in range (len(Y)):
        X.append(0)

    a=len(Y)-1
    X[a]=Y[a]
    for i in range (1,len(Y)):

        soma1 = 0
        for j in range (len(Y), (a-i), -1):
            soma1 = soma1 + float(X[j-1]) * float (L[j-1][a-i])

        X[a-i] = Y[a-i] - soma1

    return X
```

Perceba que, o que denominamos de x em nosso algoritmo de resolução dos sistemas lineares é exatamente o vetor de intensidades a que procuramos encontrar através do sistema normal desenvolvido anteriormente.

5. Testes Realizados

Para todos os testes realizados, foi-se pedido para considerarmos a função r(t) = 10(1 + cos(5t)) para a intensidade e o T = 1 para a discretização temporal.

5.1. Item A

No teste A, foi utilizado para verificarmos o programa com um exemplo bem simples (sistema normal 1x1) e, inicialmente, com os parâmetros N = 128, nf =1 e $p_1 = 0,35$. Definimos a função u em $u_1(x_i) =$

$7u_1(T; x_i)$, para que possamos percorrer toda a sua malha e, consequentemente, resolver o problema inverso proposto nela.

O resultado obtido saiu de acordo com o esperado, com um sistema linear 1 x 1, de solução trivial e $a_1 = 7$. Entretanto, o python apresentou um pequeno erro residual em suas contas, da ordem de 10^{-12}, que não altera o resultado final e nem o tira do intervalo buscado.

```
Os testes requisitados para o relatório são:
A , B , C , D (digite uma dessas letras em CapsLock):
A


O teste selecionado foi o teste A
Os parametros são N = 128, nf = 1 e p1 = 0.35. Aqui o uT(xi) = 7*u1(T, xi).


O vetor coluna das intensidades encontradas foi:

[6.999999999999999]

Portanto, a1 = 7.0



O erro encontrado para este item foi:
1.1406199304247614e-15
```

5.2. Item B

Na segunda leva de testes feitos para verificação do programa, seguimos com os valores anteriormente utilizados de parâmetros. Dessa vez utilizaremos uma função mais complexa para calcularmos a solução do seu problema inverso, testando seu método e recuperando os coeficientes que representam a intensidade de suas fontes.

A função em questão será dada por: $u_T(xi) = 2.3u_1(T, x_i) + 3.7u_2(T, x_i) + 0.3u_3(T, x_i) + 4.2u_4(T, x_i)$. Neste item pudemos perceber um erro muito pequeno, da mesma ordem do item anterior, o que nos levou a um resultado satisfatório do vetor coluna encontrado para as intensidades.

```
Os testes requisitados para o relatório são:
A , B , C , D (digite uma dessas letras em CapsLock):
B


O teste selecionado foi o teste B
Os parametros são N = 128, nf = 4 e p1 = 0.15, p2 = 0.3, p3 = 0.7 e p4 = 0.8.
Aqui o uT(xi) = 2.3*u1(T, xi) + 3.7*u2(T, xi) + 0.3*u3(T, xi) + 4.2*u4(T, xi).


O vetor coluna das intensidades encontradas foi:

[2.3000000000000007, 3.699999999999993, 0.3000000000000229, 4.199999999999978]

Portanto, os valores das intensidades em 3 casas decimais são: a1 = 2.3, a2 = 3.7, a3 = 0.3, a4 = 4.2



O erro encontrado para este item foi:
5.642914415986467e-15
```

5.3. Item C

Neste terceiro item dos testes, utilizamos o arquivo em texto fornecido pelos professores no E-Disciplinas, que continha as fontes pontuais p e os valores de temperatura discretizados espacialmente para 2048 pontos. Com isso, rodamos 5 testes para diferentes valores cada valor de N (que poderão ser conferidos logo abaixo) e que apresentam resultados ligeiramente distintos conforme a quantidade de pontos da discretização espacial N são variados.

A função utilizada neste item para construir o gráfico recupera os valores de u reconstruídos pelas intensidades assim como os obtidos inicialmente pelo método de Crank-Nicholson. Disso, realizamos uma comparação, da qual podemos perceber resultados muitos similares (no gráfico, uma sobreposição quase que completa das linhas), já que as diferenças são praticamentes imperceptíveis.

Os valores do arquivo são utilizados em pontos de malha específicos quando o valor de N é menor ou igual a 2048. Nestas condições, a função de leitura de arquivos do código irá escolher os valores de Δx da malha pulando para o 2048/N número seguinte da lista.

Por exemplo: ao pegarmos um N = 512, encontramos a razão para a ordem dos pontos (2048/512 = 4), com a qual iremos pautar a escolha de pontos (...x_4, x_8, x_{12}…). Isso só não acontece com 2048, já que, neste caso, utilizamos todos os pontos disponíveis para os testes (2048/2048 = 1).

Para sermos capazes de ler o arquivo txt. do exercício, definimos uma função que o receba, abra, leia e compile seus dados. Ao retornar os valores legíveis para o programa, podemos aplicá-lo ao nosso item, utilizando seus dados para a execução da atividade.

O código utilizado para realizar este procedimento encontra-se logo abaixo:

```
# Leitura de arquivo teste.txt original
def ler_arquivo(arquivo, prox_posicao):

  uT = []
  with open(arquivo) as arq:

      linha0 = arq.readline() #Ignoramos a primeira linha que é os pnf

      linha = arq.readline()
      contador = 0
      while linha:

        # Selecionamos o xi ao depender do N escolhido
        if (contador % prox_posicao == 0):
            a = float(linha.strip())
            uT.append(a)

        linha = arq.readline()
        contador += 1

  arq.close()

  # Remoção dos valores de fronteiras
  uT.pop(-1)
  uT.pop(0)

  return uT
```

Além disso, vale ressaltar que descartamos os valores das fronteiras, já que estes são nulos e desconsiderados para o nosso experimento. Sendo assim, teremos N - 1 valores de u. Nosso objetivo neste teste se voltou para a obtenção dos valores finais das intensidades e também ao erro.

Com a função desenvolvida para calcular o erro, relacionamos nossas funções uk(T,xi) e seus coeficientes com a função principal, procurada, definindo uma diferença entre a somatória dos produtos dos primeiros e a função. Considerando essa diferença em cada ponto da malha, da forma como está descrito no próprio enunciado do exercício programa, podemos encontrar o erro ao obtermos as aproximações desejadas no programa.

Podemos, por fim, destacar o comportamento dos erros obtidos ao longo dos testes. Até o penúltimo valor de N (1024), os valores dos erros foram decrescendo consistentemente em pequenas porções, comportamento que pode ser inferido devido ao aumento de discretizações espaciais.

Em N = 2048, entretanto, há um enorme salto no decréscimo do erro, que destoa muito dos demais pelo seu tamanho reduzido. Isso ocorre pois o aumento do número de pontos utilizados, abre menos espaço para que haja a formação de erros mais amplos, configurando uma malha mais eficiente do ponto de vista experimental.

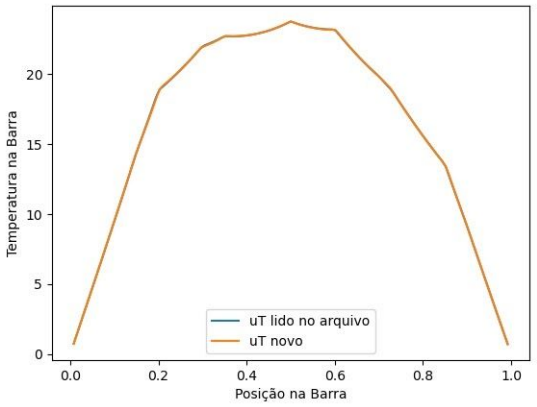
Resultados deste item:

- **N = 128**

Resultados gerais obtidos diretamente da execução do código:

```
O teste selecionado foi o teste C.
Aqui o uT(xi) foi fornecido no arquivo teste.txt da disciplina

Digite um valor de N válido (os valores são 128, 256, 512, 1024, 2048):
128

Insira o caminho para o arquivo fornecido pela disciplina (denominado teste.txt)
D:/Users/André/Desktop/MAP_EP2/teste.txt

O vetor coluna das intensidades encontradas foi:

[1.2091231792047346, 4.839258715746226, 1.8872408557571063, 1.583399931863891, 2.214504046287825, 3.1212947787782, 0.37734028636332884, 1.4923482881308132, 3.9751
388015979456, 0.40414515364879117]

Portanto, os valores das intensidades em 9 casas decimais são:

a1 = 1.209123179, a2 = 4.839258716, a3 = 1.887240856, a4 = 1.583399932, a5 = 2.214504046
a6 = 3.121294779, a7 = 0.377340286, a8 = 1.492348288, a9 = 3.975138802, a10 = 0.404145154



O erro encontrado para este item foi:
0.024453403799692568
```

Resultados gráficos:

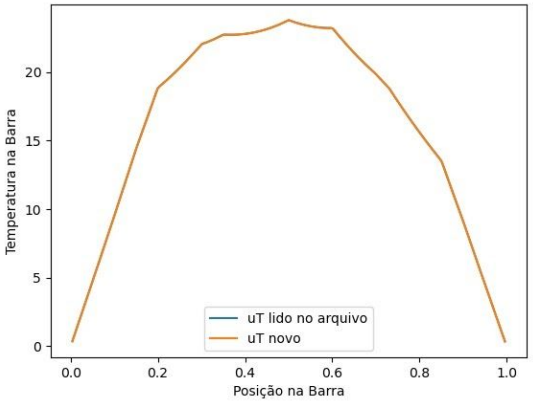


- **N = 256**

Resultados gerais obtidos diretamente da execução do código:

```
O teste selecionado foi o teste C.
Aqui o uT(xi) foi fornecido no arquivo teste.txt da disciplina

Digite um valor de N válido (os valores são 128, 256, 512, 1024, 2048):
256

Insira o caminho para o arquivo fornecido pela disciplina (denominado teste.txt)
D:/Users/André/Desktop/MAP_EP2/teste.txt

O vetor coluna das intensidades encontradas foi:

[0.9045010343172741, 5.077572635561879, 2.1008535954787533, 1.4141556850880654, 2.2292450130539514, 3.1046138569903743, 0.5094525973938095, 1.3865087904554567, 3.
949878646152565, 0.4148931283307843]

Portanto, os valores das intensidades em 9 casas decimais são:

a1 = 0.904501034, a2 = 5.077572636, a3 = 2.100853595, a4 = 1.414155685, a5 = 2.229245013
a6 = 3.104613857, a7 = 0.509452597, a8 = 1.38650879, a9 = 3.949878646, a10 = 0.414893128



O erro encontrado para este item foi:
0.012363464048876921
```

Resultados gráficos:

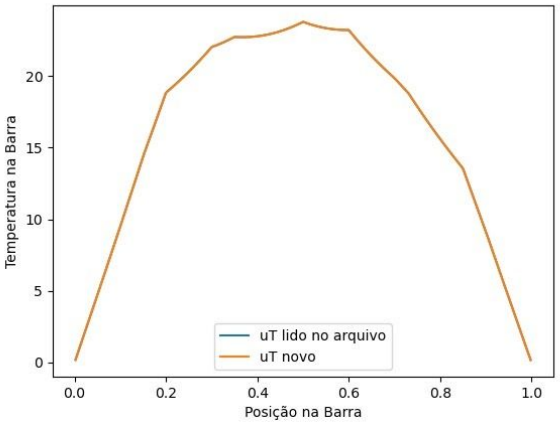


- **N = 512**

Resultados gerais obtidos diretamente da execução do código:

```
O teste selecionado foi o teste C.
Aqui o uT(xi) foi fornecido no arquivo teste.txt da disciplina

Digite um valor de N válido (os valores são 128, 256, 512, 1024, 2048):
512

Insira o caminho para o arquivo fornecido pela disciplina (denominado teste.txt)
D:/Users/André/Desktop/MAP_EP2/teste.txt

O vetor coluna das intensidades encontradas foi:

[0.9286883784938986, 5.053707844479206, 2.0437010489044596, 1.4676706728644469, 2.196763331997266, 3.09113116889867, 0.6375875163807487, 1.2716872153191225, 3.878
0948673276563, 0.5305567786424973]

Portanto, os valores das intensidades em 9 casas decimais são:

a1 = 0.928688378, a2 = 5.053707844, a3 = 2.043701049, a4 = 1.467670673, a5 = 2.196763332
a6 = 3.091131169, a7 = 0.637587516, a8 = 1.271687215, a9 = 3.878094867, a10 = 0.530556779



O erro encontrado para este item foi:
0.00847662833082880 7
```

Resultados gráficos:

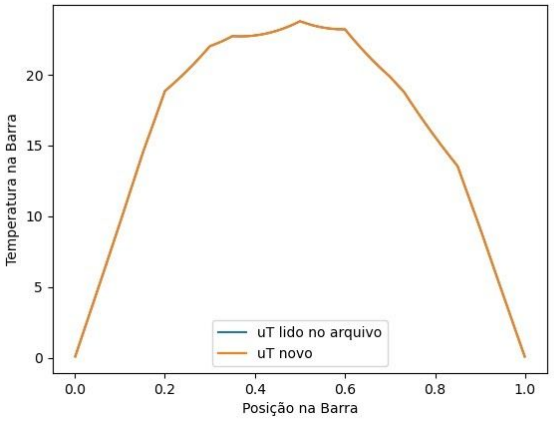


- **N = 1024**

Resultados gerais obtidos diretamente da execução do código:

```
O teste selecionado foi o teste C.
Aqui o uT(xi) foi fornecido no arquivo teste.txt da disciplina

Digite um valor de N válido (os valores são 128, 256, 512, 1024, 2048):
1024

Insira o caminho para o arquivo fornecido pela disciplina (denominado teste.txt)
D:/Users/André/Desktop/MAP_EP2/teste.txt

O vetor coluna das intensidades encontradas foi:

[1.0072813220757553, 4.992443012452448, 1.98587672761591, 1.5132584652236645, 2.1926928376831754, 3.095152875934141, 0.6523266477792884, 1.2537898890629489, 3.879
6670694049, 0.5297366253019816]

Portanto, os valores das intensidades em 9 casas decimais são:

a1 = 1.007281322, a2 = 4.992443012, a3 = 1.985876728, a4 = 1.513258465, a5 = 2.192692838
a6 = 3.095152876, a7 = 0.652326648, a8 = 1.253789889, a9 = 3.879667057, a10 = 0.529736625



O erro encontrado para este item foi:
0.0037793104632873067
```

Resultados gráficos:

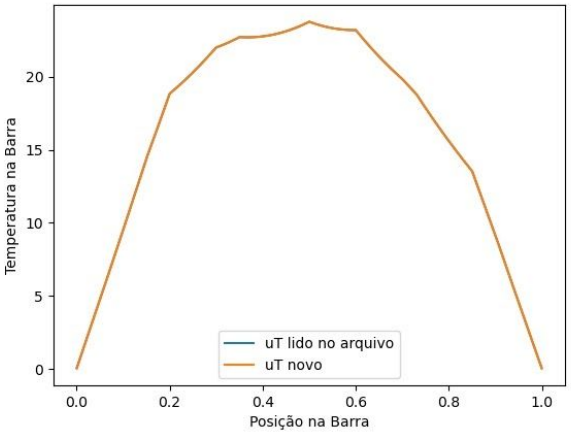


- **N = 2048**

Resultados gerais obtidos diretamente da execução do código:

```
O teste selecionado foi o teste C.
Aqui o uT(xi) foi fornecido no arquivo teste.txt da disciplina

Digite um valor de N válido (os valores são 128, 256, 512, 1024, 2048):
2048

Insira o caminho para o arquivo fornecido pela disciplina (denominado teste.txt)
D:/Users/André/Desktop/MAP_EP2/teste.txt

O vetor coluna das intensidades encontradas foi:

[0.9999999999981348, 5.000000000001982, 1.9999999999992575, 1.5000000000013927, 2.2000000000022393, 3.0999999999981753, 0.6000000000052159, 1.2999999999954674, 3.
9000000000009654, 0.49999999999974576]

Portanto, os valores das intensidades em 9 casas decimais são:

a1 = 1.0, a2 = 5.0, a3 = 2.0, a4 = 1.5, a5 = 2.2
a6 = 3.1, a7 = 0.6, a8 = 1.3, a9 = 3.9, a10 = 0.5



O erro encontrado para este item foi:
3.778137279168899e-13
```

Resultados gráficos:



5.4. Item D

Para o último item de testes do EP2, consideramos o mesmo arquivo e N's do item C), porém com algumas modificações. Com as mesmas condições pré-estabelecidas anteriormente, e também já discutidas, iremos introduzir o conceito de ruído nestes dados, o que trará um acréscimo aos erros da medição da temperatura final. Estes erros geram resultados menos precisos, já que o ruído eleva o erro total referente a cada função $u_T(x_i)$ presente.

Devido à características próprias do ruído, seus valores podem alterar dinamicamente os erros nos quais interferem. Dessa forma, utilizamos uma função random() existente em python, que produz em cada chamada do código em que está operante, um valor (pseudo) aleatório que varia entre 0 e 1. Ao subtrairmos por 0,5 e multiplicarmos por 2, obtemos um número entre -1 e 1, que será utilizado como fator representante do ruído nos resultados. Com isso, buscamos simular erros de medição que podem ocorrer em casos reais de testes.

Devemos lembrar, também, que o ruído é a simulação da imprecisão de um instrumento de medição da temperatura. Disto, temos a representação real dos erros de medição que esse elemento traz nessa última parte do exercício programa.

Logo, este item apresenta resultados distintos aos que obtivemos no item C), justamente por conta desse ruído introduzido no erro. Com essa reconstrução, podemos notar a diferença existente, mesmo que o código tenha utilizado a mesma função do item C.

Podemos notar que os gráficos obtidos neste item, apesar da semelhança com os resultados do exercício anterior como é mostrado, apresentam flutuações nos valores obtidos. Isso faz com que o gráfico não apresente um comportamento contínuo, já que o ruído altera e oscila os valores entre uma faixa que aumenta conforme o valor N aumenta.

Por fim, diferentemente do que ocorreu no item anterior, os erros deste item crescem continuamente ao longo dos testes realizados. Isso acontece pois com o aumento do valor de pontos considerados, temos uma redução da distância entre os ruídos destacados pelo programa. Como

consequência, as interpolações entre os pontos serão mais abruptas, o que trará maiores diferenças entre os valores considerados continuamente e, consequentemente, gera um erro provavelmente maior (como se trata de erros de medição aleatórios não podemos afirmar nada para 100% dos casos experimentais sempre).

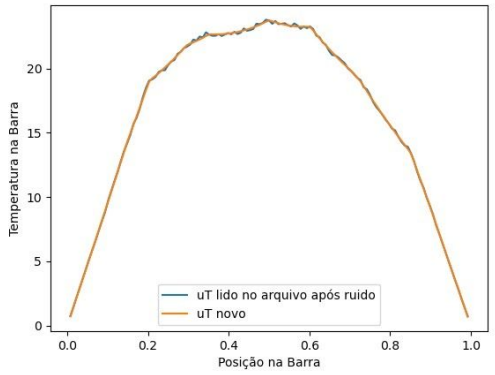
Resultados deste item:

- **N = 128**

Resultados gerais obtidos diretamente da execução do código:

```
Este teste é similar ao feito no item C, porém, acrescido da problematica de ruídos.
Digite um valor de N válido (os valores são 128, 256, 512, 1024, 2048):
128

Insira o caminho para o arquivo fornecido pela disciplina (denominado teste.txt)
D:/Users/André/Desktop/MAP_EP2/teste.txt

O vetor coluna das intensidades encontradas foi:

[1.1819225578991457, 4.961619496868138, 1.8016767875492548, 1.547589778883161, 2.165537820651023, 3.1789925376162422, 0.6537200649733732, 1.2642365094469863, 3.86
46089319481574, 0.49363913436522094]

Portanto, os valores das intensidades em 9 casas decimais são:

a1 = 1.181922558, a2 = 4.961619497, a3 = 1.801676788, a4 = 1.547589779, a5 = 2.165537821
a6 = 3.178992538, a7 = 0.653720065, a8 = 1.264236509, a9 = 3.864608932, a10 = 0.493639134



O erro encontrado para este item foi:
0.0944744218782034
```

Resultados gráficos:

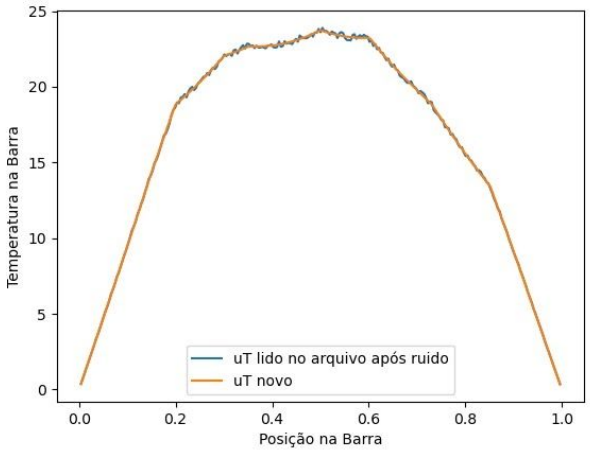


- **N = 256**

Resultados gerais obtidos diretamente da execução do código:

```
O teste selecionado foi o teste D.
Aqui o uT(xi) foi fornecido no arquivo teste.txt da disciplina

Este teste é similar ao feito no item C, porém, acrescido da problematica de ruídos.
Digite um valor de N válido (os valores são 128, 256, 512, 1024, 2048):
256

Insira o caminho para o arquivo fornecido pela disciplina (denominado teste.txt)
D:/Users/André/Desktop/MAP_EP2/teste.txt

O vetor coluna das intensidades encontradas foi:

[0.7958200467569654, 5.158758466130138, 2.1925017799182527, 1.322928254026884, 2.1443484646855886, 3.2788798497359437, 0.22393879793043148, 1.61045464439613, 3.84
056387309492, 0.5004582515882344]

Portanto, os valores das intensidades em 9 casas decimais são:

a1 = 0.795820047, a2 = 5.158758466, a3 = 2.19250178, a4 = 1.322928254, a5 = 2.144348465
a6 = 3.27887985, a7 = 0.223938798, a8 = 1.610454644, a9 = 3.840563873, a10 = 0.500458252



O erro encontrado para este item foi:
0.10518865508922386
```

Resultados gráficos:

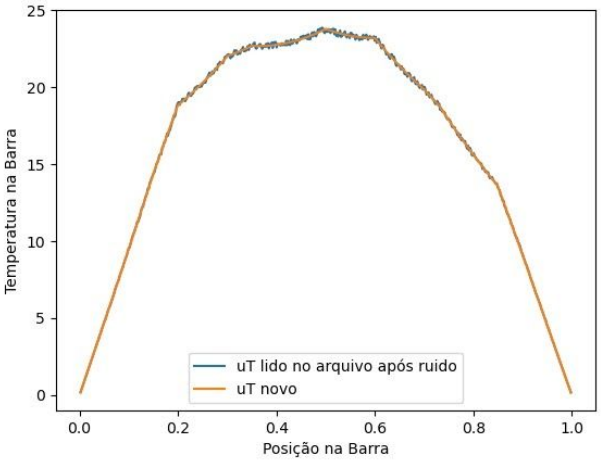


- **N = 512**

Resultados gerais obtidos diretamente da execução do código:

```
O teste selecionado foi o teste D.
Aqui o uT(xi) foi fornecido no arquivo teste.txt da disciplina

Este teste é similar ao feito no item C, porém, acrescido da problematica de ruídos.
Digite um valor de N válido (os valores são 128, 256, 512, 1024, 2048):
512

Insira o caminho para o arquivo fornecido pela disciplina (denominado teste.txt)
D:/Users/André/Desktop/MAP_EP2/teste.txt

O vetor coluna das intensidades encontradas foi:

[0.948234162705127, 5.046999972559682, 2.0797645737791406, 1.397273744594525, 2.2412442366424177, 3.072376443747067, 0.617731114675367, 1.2799356484656315, 3.9304
225913172486, 0.480739586691727]

Portanto, os valores das intensidades em 9 casas decimais são:

a1 = 0.948234163, a2 = 5.046999973, a3 = 2.079764574, a4 = 1.397273745, a5 = 2.241244237
a6 = 3.072376444, a7 = 0.617731115, a8 = 1.279935648, a9 = 3.930422591, a10 = 0.480739587



O erro encontrado para este item foi:
0.10549654072330691
```

Resultados gráficos:

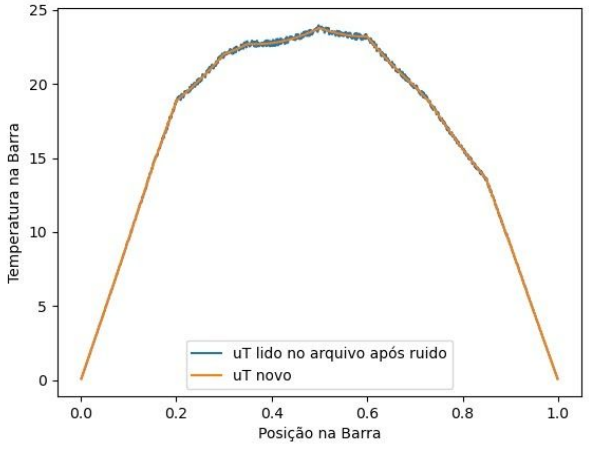


- **N = 1024**

Resultados gerais obtidos diretamente da execução do código:

```
O teste selecionado foi o teste D.
Aqui o uT(xi) foi fornecido no arquivo teste.txt da disciplina

Este teste é similar ao feito no item C, porém, acrescido da problematica de ruídos.
Digite um valor de N válido (os valores são 128, 256, 512, 1024, 2048):
1024

Insira o caminho para o arquivo fornecido pela disciplina (denominado teste.txt)
D:/Users/André/Desktop/MAP_EP2/teste.txt

O vetor coluna das intensidades encontradas foi:

[0.9817502793854374, 5.025762991104116, 2.0101969194658835, 1.4668082835257898, 2.2238363494559206, 3.090375382679306, 0.5779777516783797, 1.314794084792691, 3.91
25941898420917, 0.49172947095090536]

Portanto, os valores das intensidades em 9 casas decimais são:

a1 = 0.981750279, a2 = 5.025762991, a3 = 2.010196919, a4 = 1.466808284, a5 = 2.223836349
a6 = 3.090375383, a7 = 0.577977752, a8 = 1.314794085, a9 = 3.91259419, a10 = 0.491729471



O erro encontrado para este item foi:
0.10357507724213376
```

Resultados gráficos:

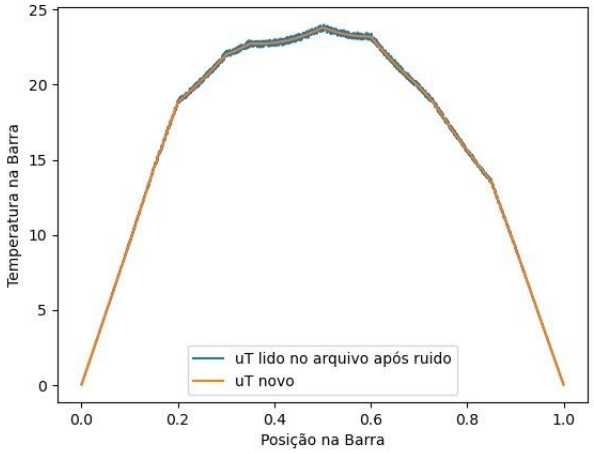


- **N = 2048**

Resultados gerais obtidos diretamente da execução do código:

```
O teste selecionado foi o teste D.
Aqui o uT(xi) foi fornecido no arquivo teste.txt da disciplina

Este teste é similar ao feito no item C, porém, acrescido da problematica de ruídos.
Digite um valor de N válido (os valores são 128, 256, 512, 1024, 2048):
2048

Insira o caminho para o arquivo fornecido pela disciplina (denominado teste.txt)
D:/Users/André/Desktop/MAP_EP2/teste.txt

O vetor coluna das intensidades encontradas foi:

[1.0226307213511845, 4.987800488397305, 1.9675203844731968, 1.5182700613263052, 2.2409956389124304, 3.032493166825417, 0.6844127709766639, 1.2499363636027745, 3.8
96218870861656, 0.5061762468326817]

Portanto, os valores das intensidades em 9 casas decimais são:

a1 = 1.022630721, a2 = 4.987800488, a3 = 1.967520384, a4 = 1.518270061, a5 = 2.240995639
a6 = 3.032493167, a7 = 0.684412771, a8 = 1.249936364, a9 = 3.896218871, a10 = 0.506176247



O erro encontrado para este item foi:
0.10476594456015416
```

Resultados gráficos:



6. Conclusão

O desenvolvimento deste exercício programa atendeu aos objetivos estabelecidos previamente, resultando em um algoritmo que, de fato, consegue computar dados computacionais e plotar gráficos com resultados satisfatórios para que estabeleçamos análises consistentes sobre o comportamento de cada método e sua aplicação. O problema inverso descrito no enunciado foi completamente desenvolvido e aplicado ao contexto da distribuição de temperatura.

As operações matemáticas na modelagem computacional não são perfeitas, existindo taxas de erros. No entanto, a acurácia da classificação como um todo foi boa, pois os erros se apresentaram consistentemente baixos em quase todos os casos, com nuances destacadas ao longo do relatório de acordo com as condições. Além disso, o desenvolvimento do exercício permitiu familiarização com o conceito de matrizes, sistemas, mínimo dos métodos quadrados e alguns métodos para resolução de diferente problemas situacionais, que, porventura, podemos nos deparar.

7. Referências

BURDEN, R. ; FAIRES, J. . Análise Numérica. São Paulo: Cengage Learning, 2008.

AL-MAMUN, A. . A study on an analytic solution 1D heat equation of a parabolic partial differential equation and implement in computer programming. International Journal of Scientific and Engineering Research Research, 2018.

CHAPRA ; CANALE. Métodos Numéricos para Engenharia. McGraw-Hill : Ed. 12, 2009

FORGER, M. . Equações Diferenciais Parciais. São Paulo: IME-USP, 2014.