Um problema inverso para obtenção de distribuição de Temperatura

Parte 1 e Parte 2 - EP - MAP3121 - Prazo de entrega: 30 / 06

May 27, 2020

Regras do Jogo

- Você deve implementar o exercício programa em C/C++ ou Python3.x
- Python:
  - Pode usar: Matplotlib, NumPy (apenas para trabalhar com aritmética de vetores, matrizes, leitura/escrita de dados), bibliotecas básicas auxiliares: sys, time, datetime, os, math.
  - **Não** pode usar: SciPy ou outras bibliotecas de algebra linear computacional
- C, C++:
  - **Não** pode usar recursos de versões além de C/C++14.
  - Pode usar qualquer biblioteca nativa do gcc/g++ (que não exiga instalção adicional).
- Incluir, obrigatoriamente, um arquivo LEIAME.txt com instruções de compilação e execução, indicando versão de interpretador/compilador necessário.
- O exercício pode ser feito em duplas, não necessariamente da mesma turma.
- Apenas um aluno deve entregar o exercício, destacando no relatório e código o nome de ambos os alunos.
- A entrega deve conter o relatório (em .pdf), contendo a análise do problema estudado, e o código usado para as simulações computacionais (arquivos fonte). A entrega deve ser feita em um arquivo compactado único.
- O relatório deve apresentar resultados e análises de todas as tarefas descritas neste enunciado.
- O seu código deve estar bem documentado, de forma a facilitar a correção. Rodar os testes também deve ser fácil para o usuário do seu programa, sem que este tenha que editar seu código. Ou seja, você deve pedir como entrada qual teste o usuário quer rodar, qual método e os parâmetros para o teste.

1 Introdução

Problemas inversos são opostos aos chamados *problemas diretos*. Basicamente, em um problema direto determina-se o efeito gerado por uma causa, enquanto que no problema inverso, a partir do efeito observado procura-se recuperar a causa. A situação mais comum que origina um problema inverso é a necessidade de interpretar medidas físicas indiretas de um objeto de interesse desconhecido. Por exemplo, na tomografia de raios-X, o problema direto é determinar as imagens que obteríamos de um corpo físico cuja estrutura interna conhecemos precisamente, usando raios-X. O problema inverso correspondente é reconstruir a estrutura interna de um corpo físico desconhecido a partir do conhecimento de imagens de raios-X tiradas de diferentes direções. Na figura 1 encontra-se um exemplo bidimensional: a fatia através de uma noz (esquerda) é a causa e a coleta de dados de raios-X (direita) é o efeito.

Problemas diretos são em geral *bem-postos*. A noção de problema bem-posto foi introduzida por Jacques Hadamard (1865-1963). Um problema é bem-posto se ele sastisfaz estas três condições:

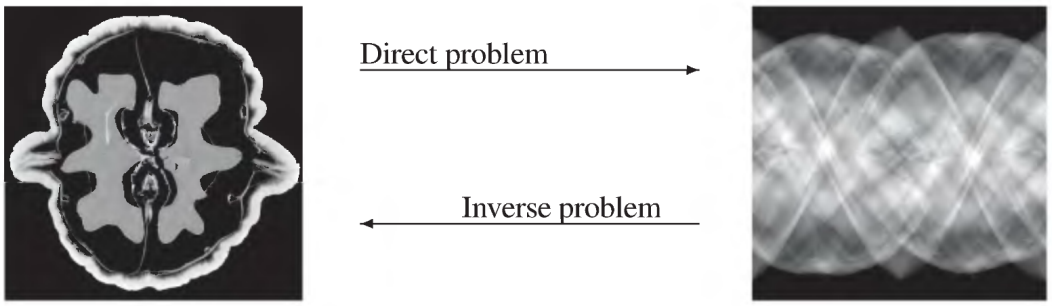

Figure 1: A imagem da fatia de uma noz à esquerda é cortesia de Keijo Hamalainen e Aki Kallonen da Universidade de Helsinque, Finlândia.

H1) Existência: existe pelo menos uma solução.

H2) Unicidade: se existir uma solução, ela é única.

H3) Estabilidade: a solução deve depender continuamente dos dados.

O exemplo típico de problema direto é uma equação diferencial parcial (EDP) da física, tais como a equação da onda ou a equação do calor. De fato, para estes problemas, conhecendo condições iniciais e de fronteiras, mais eventuais fontes, podemos calcular a solução única do problema.

Por outro lado, *problemas inversos* são frequentemente *mal-postos*, no sentido que eles não satisfazem pelo menos uma das hipóteses acima. Por exemplo, pode existir um grande número de soluções, e neste caso é dificil saber qual destas soluções é a mais relevante para a aplicação. A razão pela qual estes problemas geralmente são mal postos é porque não temos informações suficientes para encontrar a causa do efeito que estamos observando. As razões para tal são diversas, incluindo os custos de aquisição dos dados. Notemos ainda que medições em geral trazem imprecisões, o que mostra a importância da condição $H3$ de um problema bem posto. Problemas inversos são muito relevantes em aplicações na física e engenharia, quando queremos determinar parâmetros que não podemos observar diretamente. Exemplos relevantes encontram-se, por exemplo, em detecção da origem de terremotos, mapeamento de camadas geológicas para determinação de onde se encontra petróleo e em aerodinâmica, na indústria automobilística e aeroespacial.

Nesse EP, dividido em duas partes, vamos resolver um problema relativamente simples, ligado à equação do calor. Esta primeira parte contemplada no EP1, refere-se ao problema direto. Veremos como determinar a evolução da distribuição de temperatura em uma barra sujeita a fontes de calor, a partir de uma dada distribuição inicial. No EP2 a partir da solução em um instante T final, iremos determinar a intensidade das fontes de calor. Mantemos aqui ainda o enunciado do EP1, pois você irá usar partes do programa já desenvolvido na solução deste novo problema. Vá então para a seção que descreve o EP2 3.

2 Descrição do problema direto - equação do calor

A evolução da distribuição de temperatura em uma barra é dada pela seguinte equação diferencial parcial:

$$u_t(t,x) = u_{xx}(t,x) + f(t,x) \text{ em } [0,T] \times [0,1], \tag{1}$$
$$u(0,x) = u_0(x) \text{ em } [0,1] \tag{2}$$
$$u(t,0) = g_1(t) \text{ em } [0,T] \tag{3}$$
$$u(t,1) = g_2(t) \text{ em } [0,T]. \tag{4}$$

Aqui, t é a variável temporal e x a variável espacial. Estamos usando uma notação compacta para as derivadas parciais. Por exemplo,

$$u_{xx}(t,x) = \frac{\partial^2 u(t,x)}{\partial x^2} \ .$$

Normalizamos o comprimento da barra para 1 e vamos integrar a equação num intervalo de tempo de 0 a T. A variável $u(t,x)$ descreve a temperatura no instante t na posição x, sendo a distribuição inicial $u_0(x)$ dada. Na descrição acima as condições de fronteira (3)-(4) são do tipo Dirichlet, com as temperaturas

nos extremos da barra prescritas. Alternativamente poderia ser prescrito o fluxo de calor nos extremos, com as derivadas de u dadas. A função f descreve as fontes de calor ao longo do tempo.

2.1 Discretizações da equação do calor

Queremos aproximar numericamente a solução de (1)-(4). Uma forma simples para se obter uma aproximação numérica das derivadas parciais é aproximá-las por *diferenças finitas*. Estas são baseadas em expansões de Taylor. Por exemplo, se $g \in C^k(a,b)$ e $x \in (a,b)$ temos que:

$$g(x+h) = g(x) + g'(x)h + g''(x)h^2/2 + \cdots + g^{(k-1)}(x)h^{k-1}/(k-1)! + g^{(k)}(\bar{x})h^k/k! \ ,$$

onde $\bar{x}$ é um ponto entre x e $x+h$. Usando combinações desta expressão obtemos aproximações para os valores de uma função e suas derivadas. Por exemplo, temos:

$$g(x) = \frac{g(x-h)+g(x+h)}{2} + O(h^2) \tag{5}$$

$$g'(x) = \frac{g(x+h)-g(x)}{h} + O(h) \tag{6}$$

$$g'(x) = \frac{g(x+h)-g(x-h)}{2h} + O(h^2) \tag{7}$$

$$g''(x) = \frac{g(x+h)-2g(x)+g(x-h)}{h^2} + O(h^2) \tag{8}$$

onde a notação $O(h^k)$ denota um erro proporcional a h^k. O termo exato depende de derivadas de ordem mais alta da função. Para a aproximação de derivadas parciais em relação a uma das variáveis usamos as mesmas fórmulas, com relação à variável em consideração. Por exemplo:

$$u_t(t,x) = \frac{u(t+\Delta t,x)-u(t,x)}{\Delta t} - \Delta t \frac{u_{tt}(\bar{t},x)}{2} \text{ e} \tag{9}$$

$$u_{xx}(t,x) = \frac{u(t,x-\Delta x)-2u(t,x)+u(t,x+\Delta x)}{\Delta x^2} - \Delta x^2 \frac{u_{xxxx}(t,\bar{x})}{4!} \tag{10}$$

onde $\bar{t}$ é um valor entre t e $t+\Delta t$ e $\bar{x}$ é um valor entre $x-\Delta x$ e $x+\Delta x$. Nas expressões para $u_t(t,x)$ e $u_{xx}(t,x)$ dadas acima os erros são proporcionais a Δt e Δx^2 respectivamente. Se estes incrementos tenderem a zero teremos convergência das aproximações por diferenças finitas para as derivadas parciais correspondentes.

Para a discretização da equação do calor vamos introduzir uma malha espacial dada pelos pontos $x_i = i\Delta x,\ i = 0, \cdots, N$, com $\Delta x = 1/N$. Para a discretização temporal definimos $\Delta t = T/M$, e calculamos aproximações nos instantes $t_k = k\Delta t,\ k = 1, \cdots, M$. Denotamos a aproximação para a solução nos pontos de malha $u(t_k, x_i)$ por u_i^k.

Desta forma teremos a condição inicial dada por:

$$u_i^0 = u_0(x_i), i = 0, \cdots, N$$

Ao longo da evolução temporal as condições de fronteira são dadas por

$$u_0^k = g_1(t_k) \text{ e } u_N^k = g_2(t_k),\ k = 1, \cdots, M \ .$$

Para os pontos interiores a evolução é aproximada pelas fórmulas de diferenças finitas:

$$u_i^{k+1} = u_i^k + \Delta t\left(\frac{u_{i-1}^k - 2u_i^k + u_{i+1}^k}{\Delta x^2} + f(x_i,t_k)\right) , \quad i = 1, \cdots, N-1, \text{ e } k = 0, \cdots, M-1 \ . \tag{11}$$

Sabendo os valores iniciais da temperatura e seus valores na fronteira ao longo do tempo, a expressão (11) permite facilmente a determinação da solução aproximada em todos os instantes, computando sequencialmente desde $t_0 = 0$ a $t_M = T$.

2.2 Um pouco de teoria

Seja $u(t,x)$ a solução exata da equação do calor (1)-(4). Definimos o erro local de truncamento, que mede quão bem a solução exata $u(t,x)$ satisfaz à equação discretizada (11) (na forma dividida por Δt), dado por

$$\tau_i^k(\Delta t, \Delta x) = \frac{u(t_{k+1}, x_i) - u(t_k, x_i)}{\Delta t} - \frac{u(t_k, x_{i-1}) - 2u(t_k, x_i) + u(t_k, x_{i+1})}{\Delta x^2} - f(x_i, t_k) \tag{12}$$

$$= u_t(t_k, x_i) + \Delta t \frac{u_{tt}(\bar{t_k}, x_i)}{2} - u_{xx}(t_k, x_i) - \Delta x^2 \frac{u_{xxxx}(t_k, \bar{x_i})}{4!} - f(x_i, t_k) \tag{13}$$

$$= \Delta t \frac{u_{tt}(\bar{t_k}, x_i)}{2} - \Delta x^2 \frac{u_{xxxx}(t_k, \bar{x_i})}{4!} \ , \tag{14}$$

onde usamos as expansões de Taylor (9) e (10) para as aproximações das derivadas parciais e o fato de $u(t,x)$ ser a solução exata da equação do calor na última passagem. Com a hipótese de $u(t,x)$ ter 4 derivadas contínuas em x e duas em t no domínio de integração $D = [0,T] \times [0,1]$ podemos delimitar

$$\tau(\Delta t, \Delta x) = \max_{k,i} |\tau_i^k(\Delta t, \Delta x)| \leq C_1 \Delta t + C_2 \Delta x^2 \tag{15}$$

com

$$C_1 = \max_D |\frac{u_{tt}(t,x)}{2}| \text{ e } C_2 = \max_D |\frac{u_{xxxx}(t,\bar{x})}{4!}| \ .$$

Como consequência temos que

$$\lim_{\Delta t, \Delta x \to 0} \tau(\Delta t, \Delta x) = 0 \tag{16}$$

Convergência da solução

Vimos que sob razoáveis hipóteses temos que o erro local de truncamento converge a zero. No entanto, o que realmente desejamos é que a solução aproximada nos pontos de malha convirja para a solução exata da equação do calor. Vamos agora estabelecer uma condição que garante que isto ocorre. Inicialmente definimos o erro entre a solução aproximada e a exata como:

$$e_i^k = u(t_k, x_i) - u_i^k \ . \tag{17}$$

Podemos agora combinar a equação para a determinação da solução aproximada (11) e a equação definindo o erro local de truncamento (12) para obter a seguinte equação para o erro:

$$e_i^{k+1} = e_i^k + \Delta t(\frac{e_{i-1}^k - 2e_i^k + e_{i+1}^k}{\Delta x^2} + \tau_i^k)) \ , \quad i = 1, \cdots, N-1, \text{ e } k = 0, \cdots, M-1 \ . \tag{18}$$

Vamos ainda definir a norma do erro no instante t_k como

$$||e^k|| = \max_i |e_i^k| \ . \tag{19}$$

Vamos agora estimar como o erro evolui em função do tempo. Vamos denominar $\lambda = \Delta t / \Delta x^2$. Então da equação do erro (18) obtemos que

$$|e_i^{k+1}| \leq |1 - 2\lambda||e_i^k| + |\lambda|(|e_{i-1}^k| + |e_{i+1}^k|) + \Delta t|\tau_i^k| \tag{20}$$

$$\leq (|1 - 2\lambda| + 2|\lambda|)||e^k|| + \Delta t \tau(\Delta t, \Delta x) \tag{21}$$

Temos que λ é sempre positivo. Por outro lado, se $\lambda \leq 1/2$ então $1 - 2\lambda \geq 0$. Assim, para esta escolha de λ, obtemos

$$|e_i^{k+1}| \leq ((1 - 2\lambda) + 2\lambda)||e^k|| + \Delta t \tau(\Delta t, \Delta x) \tag{22}$$

$$\leq ||e^k|| + \Delta t \tau(\Delta t, \Delta x), \text{ e segue que } ||e^{k+1}|| \leq ||e^k|| + \Delta t \tau(\Delta t, \Delta x) \ . \tag{23}$$

Usando esta última estimativa recursivamente obtemos então que:

$$||e^{k+1}|| \leq ||e^{k-1}|| + 2\Delta t \tau(\Delta t, \Delta x) \tag{24}$$

$$\leq ||e^0|| + (k+1)\Delta t \tau(\Delta t, \Delta x) = t_{k+1} \tau(\Delta t, \Delta x) \tag{25}$$

$$\leq T(C_1 \Delta t + C_2 \Delta x^2) \ . \tag{26}$$

Nesta última estimativa usamos ainda que o erro inicial $||e^0||$ é nulo, uma vez que $u_0(x)$ é dado, e a delimitação do erro de truncamento (15). Esta estimativa mostra, que se Δt e Δx tenderem a zero o erro também vai a zero e portanto a aproximação calculada converge para a solução exata da equação. Note no entanto, que nesta demonstração usamos fortemente a hipótese de que

$$\lambda = \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2} \, . \tag{27}$$

Caso esta condição seja violada, você verificará com seu programa, o método fica instável, com o erro se amplificando enormemente. Este método é dito *condicionalmente convergente*. Para termos convergência, Δt deve ser da ordem de Δx^2. Portanto, se a condição (27) estiver satisfeita para $\Delta t = \alpha \Delta x^2$, com $\alpha \leq 1/2$, obtemos então que o erro fica menor que uma constante vezes Δx^2. Dizemos que o método é (condicionalmente) convergente de ordem 2 em Δx.

2.3 Primeira tarefa

Você deve implementar o método (11), deixando os valores de N e M (que determinam Δt e Δx) como variáveis a serem escolhidas em tempo de execução. Teste o seu programa com os dados
a) $T = 1$ com a fonte $f(t,x) = 10x^2(x-1) - 60xt + 20t$ a partir de $u_0(x) = 0$ e condições de fronteira também nulas. Faça integrações com $N = 10, 20, 40, 80, 160$ e 320 para $\lambda = 0.5$ e $\lambda = 0.25$. Experimente também com $\lambda = 0.51$. O que acontece? Verifique que a solução exata neste caso é igual a $u(t,x) = 10tx^2(x-1)$ e calcule o erro obtido em $T = 1$ com as diversas resoluções para $\lambda = 0.5$ e $\lambda = 0.25$. Verifique o comportamento do erro. Qual o fator de redução esperado a cada refinamento de malha. Qual o número de passos necessários ao se usar $N = 640$? E se dobrarmos N?

Observação: Calcule o erro de truncamento para esse exemplo e constate que é igual a zero! Neste caso seu código deve gerar a solução exata, com erro nulo em T=1 (a menos de erros de arredondamento, ou seja não vai dar zero exatamente ...). Inclusive, ao refinar a malha e fazer mais passos este erro de arredondamento tende a crescer. Ok, agora que você já verificou este fato, vamos mudar a função f do item a) para

$$f(t,x) = 10cos(10t)x^2(1-x)^2 - (1 + sin(10t))(12x^2 - 12x + 2)$$

que corresponde à solução exata $u(t,x) = (1 + sin(10t))x^2(1-x)^2$, com valor inicial $u_0(x) = x^2(1-x)^2$ e condições nulas na fronteira. Os testes descritos neste item, obrigatórios para entregar, devem ser feitos com esta última função. Quem quiser incluir também a primeira em seu relatório, também pode.

b) Determine quem deve ser $u_0(x)$, $g_1(t)$, $g_2(t)$ e $f(t,x)$ de forma que a solução exata seja dada por $u(t,x) = e^{t-x}\cos(5tx)$ e repita os experimentos da parte a)

c) $T = 1$ a partir da condição inicial nula, com uma fonte pontual localizada em um ponto p do domínio e intensidade variando ao longo do tempo dada por $r(t) = 10000 * (1 - 2t^2)$. Vamos agora ver como implementar uma fonte pontual, digamos de intensidade 1. Podemos ver esta força f localizada, como limite de forças g_h que atuam em uma pequena região (cujo tamanho vai a zero com h) em torno do ponto em questão (ou seja, são não nulas apenas nesta pequena região), de forma que a integral de cada g_h seja constante igual a 1. Para obter tal efeito podemos adotar

$$g_h(x) = \frac{1}{h}, \text{ se } p - h/2 \leq x \leq p + h/2, \text{ e } g_h(x) = 0 \text{ caso contrário.}$$

Alternativamente $g_h(x)$ poderia assumir o valor $1/h$ em p e variar linearmente de 0 a $1/h$ no intervalo $[p-h, p]$ e de $1/h$ a 0 no intervalo $[p, p+h]$, sendo nula no restante do domínio. Adote uma destas duas representações de g_h e defina a fonte pontual $f(t,x) = r(t)g_h(x)$, com $h = \Delta x$.

Trabalhe com a fonte em $p = 0.25$ e use na fronteira $g_1(t) = g_2(t) = 0$. Neste caso não sabemos a solução exata. Determine-a numericamente.

Em todos os casos faça gráficos da solução obtida. Plotem os gráficos de 0.1 em 0.1, assim podem observar a evolução temporal.

2.4 Um método implícito

Você deve ter percebido através de seus experimentos com o método (11) que a quantidade de passos no tempo, necessários para se fazer integrações em malhas mais finas, cresce muito. Isto faz com que o

esquema não seja muito eficiente. O que gostaríamos é ter um método convergente de ordem 2 em que pudéssemos usar Δt da mesma ordem que Δx. Para um tal método ser estável (você já deve ter notado o que ocorre quando há instabilidade ...), será no entanto necessário usar um método implícito. No esquema (11) a solução em cada ponto em um novo instante de tempo é obtida simplesmente avaliando uma combinação de valores vizinhos do passo anterior. Em um método implícito, a solução em um ponto de malha no novo instante depende também de outros valores no mesmo instante. Esta interdependência dos valores no novo instante leva à necessidade de resolver um sistema de equações a cada passo no tempo. Um primeiro exemplo de método implícito (também chamado de Euler implícito) é dado pelo esquema

$$u_i^{k+1} = u_i^k + \lambda(u_{i-1}^{k+1} - 2u_i^{k+1} + u_{i+1}^{k+1}) + \Delta t f(x_i, t_{k+1}) \ , \quad i = 1, \cdots, N-1, \text{ e } k = 0, \cdots, M-1 \ , \tag{28}$$

com as mesmas notações anteriores. Neste método, para a evolução temporal, necessitamos resolver a cada passo um sistema linear com uma matriz A tridiagonal simétrica, como segue

$$\begin{bmatrix} 1+2\lambda & -\lambda & 0 & \cdots & 0 \\ -\lambda & 1+2\lambda & -\lambda & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -\lambda & 1+2\lambda & -\lambda \\ 0 & \cdots & 0 & -\lambda & 1+2\lambda \end{bmatrix} \begin{bmatrix} u_1^{k+1} \\ u_2^{k+1} \\ \vdots \\ u_{N-2}^{k+1} \\ u_{N-1}^{k+1} \end{bmatrix} = \begin{bmatrix} u_1^k + \Delta t f_1^{k+1} + \lambda g_1(t^{k+1}) \\ u_2^k + \Delta t f_2^{k+1} \\ \vdots \\ u_{N-2}^k + \Delta t f_{N-2}^{k+1} \\ u_{N-1}^k + \Delta t f_{N-1}^{k+1} + \lambda g_2(t^{k+1}) \end{bmatrix} \tag{29}$$

Analogamente ao que fizemos para o método explícito (11) definimos o erro de truncamento

$$\tau_i^k(\Delta t, \Delta x) = \frac{u(t_{k+1}, x_i) - u(t_k, x_i)}{\Delta t} - \frac{u(t_{k+1}, x_{i-1}) - 2u(t_{k+1}, x_i) + u(t_{k+1}, x_{i+1})}{\Delta x^2} - f(x_i, t_{k+1}) \tag{30}$$

e a delimitação

$$\tau(\Delta t, \Delta x) = \max_{k,i} |\tau_i^k(\Delta t, \Delta x)| \leq C_1 \Delta t + C_2 \Delta x^2 \ . \tag{31}$$

Procedendo como anteriormente chegaremos à expressão (verifique!):

$$(1+2\lambda)|e_i^{k+1}| \leq |e_i^k| + \lambda(|e_{i-1}^{k+1}| + |e_{i+1}^{k+1}|) + \Delta t|\tau_i^k| \tag{32}$$

$$\leq ||e^k|| + 2\lambda||e^{k+1}|| + \Delta t \tau(\Delta t, \Delta x) \ , \tag{33}$$

de onde segue que

$$||e^{k+1}|| = (1 + 2\lambda - 2\lambda)||e^{k+1}|| \leq ||e^k|| + \Delta t \tau(\Delta t, \Delta x) \ . \tag{34}$$

Esta equação é análoga à (22), porém foi obtida sem que precisássemos fazer qualquer restrição na escolha de Δt e Δx. A demonstração da convergência segue daqui como feita para o método explícito. O método de Euler implícito é incondicionalmente estável e convergente de ordem 2 em Δx e ordem 1 em Δt. Assim, mesmo não sofrendo restrições de estabilidade, a precisão do esquema estará limitada pela escolha de Δt.

A seguir apresentamos o método de Crank-Nicolson, que também é incondicionalmente estável, mas tem convergência de ordem 2 em Δx e Δt.

O método é da forma:

$$u_i^{k+1} = u_i^k + \frac{\lambda}{2}\left((u_{i-1}^{k+1} - 2u_i^{k+1} + u_{i+1}^{k+1}) + (u_{i-1}^k - 2u_i^k + u_{i+1}^k)\right) + \frac{\Delta t}{2}(f(x_i, t_k) + f(x_i, t_{k+1})) \tag{35}$$

com as mesmas notações anteriores.

Este é um esquema de segunda ordem, centrado no tempo intermediário $t^k + 0.5\Delta t$, combinando as fórmulas (5), (7) e (8). Procure analisar o erro de truncamento!

Analogamente ao método implícito temos que resolver um sistema linear. Este envolverá também uma matriz tridiagonal simétrica, como no método anterior, trocando λ por $\lambda/2$. O lado direito é distinto. Será sua tarefa elaborar os detalhes.

A análise da convergência deste esquema é no entanto mais complexa que a anterior, envolvendo os auto-valores da Matriz do sistema. Esta pode ser encontrada detalhada no livro **Analysis of Numerical Methods, E. Isaacson and H. B. Keller** e não será repetida aqui.

2.5 Segunda tarefa

a) Nos dois métodos implícitos apresentados, há a necessidade de resolução de um sistema tridiagonal simétrico. Para tanto você deve escrever uma rotina que calcula a decomposição $A = LDL^t$ da matriz do sistema tridiagonal em questão, onde a matriz L é bidiagonal triangular unitária inferior e D diagonal, ou seja

$$A = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_2 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ 0 & 0 & l_{N-1} & 1 \end{bmatrix} \begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & d_{N-1} \end{bmatrix} \begin{bmatrix} 1 & l_2 & 0 & 0 \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & 1 & l_{N-1} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \tag{36}$$

Note que a matriz A, sendo tridiagonal simétrica $N-1 \times N-1$ pode ser armazenada em apenas dois vetores de comprimento $N-1$, um para a diagonal de A e o outro para a subdiagonal. Já as matrizes L e D necessitam apenas de um vetor cada uma para seu armazenamento. Seu programa para o cálculo desta decomposição deve fazer uso deste tipo de armazenamento, tendo como entrada os dois vetores representando A e como saída os dois vetores representando L e D. Note que a matriz A não se altera em função do passo no tempo, permanecendo constante ao longo de toda a integração temporal. Assim, basta calcular sua decomposição uma única vez. Escreva então um procedimento que dada uma decomposição deste tipo e um lado direito do sistema $Ax = LDL^t x = b$, compute a solução x. Você irá usar estas rotinas tanto para o Método de Euler implícito como para o de Crank-Nicolson.
b) Repita os mesmos testes da primeira tarefa com o método de Euler implícito, utilizando $\Delta t = \Delta x$. Verifique a ordem de convergência.
c) Faça o mesmo que no item anterior para o método de Crank-Nicolson.

3 Parte 2 - Um problema inverso para a equação do calor

Na segunda parte deste exercício o objetivo será, a partir do conhecimento da distribuição final de temperatura no instante T, determinar a intensidade das fontes de calor aplicadas em posições conhecidas da barra. Mais precisamente, seja $u_T(x) = u(T, x)$ a solução da equação do calor dada por (1)-(4), com condições iniciais e de fronteira nulas ($u_0(x) = g_1(t) = g_2(t) = 0$), com o termo forçante da forma:

$$f(t, x) = r(t) \sum_{k=1}^{nf} a_k g_h^k(x) \quad , \tag{37}$$

com $g_h^k(x)$ sendo forçantes pontuais em $0 < p_1 < p_2 < \cdots < p_{nf} < 1$ ao longo da barra, dadas por

$$g_h^k(x) = \frac{1}{h}, \text{ se } p_k - h/2 \leq x \leq p_k + h/2, \text{ e } g_h(x) = 0 \text{ caso contrário.}$$

A função $r(t)$ descreve uma variação temporal das forçantes e os coeficientes a_k as respectivas intensidades. Nosso problema será a determinação das intensidades a_k a partir do conhecimento de $u_T(x)$ em pontos de uma malha $x_i = i\Delta x,\ i = 0, \cdots, N$, com $\Delta x = 1/N$.

Para tanto iremos inicialmente determinar funções $u_k(t, x), k = 1, \cdots, nf$, soluções de (1)-(4), com forçantes $f_k(t, x) = r(t) g_h^k(x)$ respectivamente. Devido à linearidade das equações (com condições iniciais e de contorno nulas), teremos necessariamente que (verifique):

$$u_T(x) = \sum_{k=1}^{nf} a_k u_k(T, x) \ . \tag{38}$$

Como conheceremos apenas os valores de $u_T(x)$ medidos nos pontos x_i da malha, iremos determinar os valores de intensidade a_k de forma a minimizar

$$E_2 = \sqrt{\Delta x \sum_{i=1}^{N-1} \left(u_T(x_i) - \sum_{k=1}^{nf} a_k u_k(T, x_i) \right)^2} \ , \tag{39}$$

que corresponde a um problema de mínimos quadrados. Como as funções $u_k(T, x)$ são desconhecidas, iremos aproximar estes valores resolvendo as equações (1)-(4) através do método de Crank-Nicolson, desenvolvido no EP1 (tome sempre $M = N$ para definir Δt).

Observação: A definição do erro E_2 acima não incluiu os extremos do intervalo, pois as funções $u_T(x)$ e $u_k(T,x)$ aí se anulam, devido às condições de fronteira da equação. Este erro corresponde a uma versão discreta do erro quadrático $E = \sqrt{\int_0^1 [u_T(x) - \sum_{k=1}^{nf} a_k u_k(T,x)]^2 \, dx}$, com a integral aproximada pelo método dos trapézios, que será visto no curso.

No problema de mínimos quadrados (39) queremos aproximar o vetor de medições $u_T(x_i), i = 1, ..., N-1$ por uma combinação linear dos vetores $u_k(T,x_i), i = 1, ..., N-1$, obtidos pelas integrações usando o método de Crank-Nicolson (com $M = N$). Para a solução do problema de mínimos quadrados devemos resolver o sistema normal

$$\begin{bmatrix} < u_1, u_1 > & < u_2, u_1 > & \cdots & < u_{nf}, u_1 > \\ < u_1, u_2 > & < u_2, u_2 > & \cdots & < u_{nf}, u_2 > \\ \vdots & \vdots & \vdots & \vdots \\ < u_1, u_{nf} > & < u_2, u_{nf} > & \cdots & < u_{nf}, u_{nf} > \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{nf} \end{bmatrix} = \begin{bmatrix} < u_T, u_1 > \\ < u_T, u_2 > \\ \vdots \\ < u_T, u_{nf} > \end{bmatrix} \tag{40}$$

com o produto interno $< u, v >= \sum_{i=1}^{N-1} u(x_i)v(x_i)$.

3.1 Suas tarefas

a) Desenvolver um código que: dados os pontos $p_1, \cdots, p_{nf}$ gere os vetores $u_k(T,x_i), i = 1, ..., N-1$ a partir da integração da equação do calor (1)-(4), com condição inicial $u_0(x) = 0$ e de fronteira $g_1(t) = g_2(t) = 0$, com forçante $f(t,x) = r(t)g_h^k(x)$, $k = 1, \cdot, nf$. Para tanto você deve utilizar o método de Crank-Nicolson desenvolvido no EP1 (com $M = N$).
b) Dados os valores de $u_T(x_i), i = 1, ..., N-1$, monte a matriz e o sistema normal do problema de mínimos quadrados para o cálculo das intensidades.
c) Escreva uma rotina para calcular a decomposição LDL^t de uma matriz simétrica e outra para, dada esta decomposição, resolver um sistema linear associado à matriz decomposta. Use estas rotinas para resolver o problema de mínimos quadrados. Note que no presente problema a matriz A não será esparsa. (veja seção 6.6 do livro do Burden / Faires)

3.2 Testes

Em todos os testes utilizaremos $T = 1$ e $r(t) = 10(1 + \cos(5t))$.
a) Para os parâmetros $N = 128$, $nf = 1$ e $p_1 = 0.35$ você deve fazer uma primeira verificação do seu programa. Construa $u_1(T,x_i), i = 1, ..., N-1$. Defina $u_T(x_i) = 7u_1(T,x_i)$ e resolva o problema inverso. Neste caso, o sistema linear é 1×1, com solução trivial e você deve obter $a_1 = 7$.
b) Como segundo teste de verificação, ainda com $N = 128$, tomemos $nf = 4$ e $p_1 = 0.15$, $p_2 = 0.3$, $p_3 = 0.7$ e $p_4 = 0.8$. Construa as funções $u_k(T,x)$ e defina $u_T(x_i) = 2.3u_1(T,x_i) + 3.7u_2(T,x_i) + 0.3u_3(T,x_i) + 4.2u_4(T,x_i)$ e resolva o problema inverso, testando também seu método que resolve o sistema linear. Os coeficientes representando as intensidades das fontes devem ser recuperados.
c) No arquivo teste.txt fornecido são dadas as localizações de 10 fontes e a seguir os valores de $u_T(x)$ em uma malha com $\Delta x = 1/2048$. Estão fornecidos os valores de $u_T(x_i), i = 0, \cdots, 2048$, incluindo os extremos do intervalo. Com estes dados você deve rodar 5 testes, para os valores de $N = 128, 256, 512, 1024$ e 2048. Para os testes com N menor que 2048, você deve utilizar os valores do arquivo nos pontos de malha adequados. Por exemplo, se $N = 512$, os pontos seriam $x_0, x_4, x_8, \cdots, x_{2048}$, ou seja, se tomam os valores do arquivo de 4 em 4. Os valores nos extremos, que correspondem às fronteiras ainda serão descartados, para se ficar com $N - 1 = 511$ valores de $u_T(x)$ nesta malha. (O procedimento para os outros valores de N é análogo). Para cada N, você deve imprimir além dos valores das intensidades a_k, o valor do erro quadrático E_2, conforme (39).
d) Caso com ruído: Você deve agora repetir os testes do item c), mas com a introdução de ruído nos dados, representando erros de medição na temperatura final. Para tal, multiplique cada valor de $u_T(x_i)$ por $1. + r\epsilon$ com $\epsilon = 0.01$ e r um número randômico entre -1 e 1.

Observação: Em python a função random() (do módulo random) produz a cada chamada um valor (pseudo) aleatório entre 0 e 1. Subtraindo 0.5 e multiplicando por 2, você obterá números entre -1 e 1.
Observações finais: Seu programa deve possuir flags para que o usuário escolha qual dos casos (a,b,c,d) deseja rodar. Nos casos c) e d) deve então ser perguntado qual o valor de N escolhido. O programa deve imprimir as intensidades e o erro quadrático. Em seu relatório, além destes dados referentes a cada teste, você deve incluir uma análise dos resultados, especialmente dos itens c) e d). Inclua também gráficos da solução no instante $T = 1$ (da proveniente do arquivo e da que você obteve com os coeficientes das intensidades recuperadas). Não deixe de comentar seu código!