

Um problema inverso na geofísica.

Exercício Computacional
MAP3122 - Quadrimestral 2020
Prof. Antoine Laurain

Este exercício computacional pode ser feito em duplas. Veja as instruções detalhadas no final do texto.

1 Introdução

Problemas inversos são opostos aos *problemas diretos*. Informalmente, em um problema direto encontra-se um efeito de uma causa e , em um problema inverso, recebe-se o efeito e desejamos recuperar a sua 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

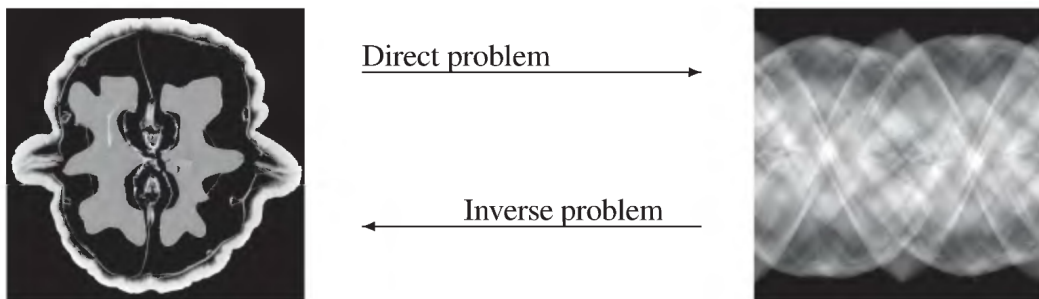


Figura 1: A imagem da fatia de um noz à esquerda é cortesia de Keijo Hamalainen e Aki Kallonen da Universidade de Helsinque, Finlândia (essa imagem vem de [1])

coleta de dados de raios-X (direita) é o efeito. Os dados tomográficos são mostrados na forma tradicional de sinograma.

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 satisfaz estas três condições:

- 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 as condições iniciais e as 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 difícil 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. Esta falta de informação pode ter muitos motivos. Muitas vezes, é porque só podemos realizar um número limitado de medições: pode ser porque essas medições são caras ou porque a região onde é possível fazer medições é pequena. Mesmo se tivermos apenas dados parciais, gostaríamos de encontrar uma solução aproximada do problema inverso. Problemas inversos são alguns dos problemas matemáticos mais importantes da ciência, da engenharia, e da física, pois nos dizem sobre parâmetros que não podemos observar diretamente.

Nesse EP vamos resolver uma versão simplificada de um problema inverso altamente relevante da geofísica. Ele tem aplicações-chaves na indústria e na geofísica, tais como o mapeamento do subsolo para prospecção de petróleo e a detecção da origem de terremotos.

2 Descrição do problema direto

Consideramos a equação diferencial parcial seguinte:

$$\partial_{tt}^2 u(t, x) - c^2 \partial_{xx}^2 u(t, x) = f(t, x) \text{ em } [0, T] \times [0, 1], \quad (1)$$

$$u(0, x) = u_0(x) \text{ em } [0, 1] \quad (2)$$

$$\partial_t u(0, x) = u_1(x) \text{ em } [0, 1] \quad (3)$$

$$u(t, 0) = 0 \text{ em } [0, T] \quad (4)$$

$$u(t, 1) = 0 \text{ em } [0, T]. \quad (5)$$

Aqui, t é a variável de tempo e x a variável de espaço, e usamos as notações

$$\partial_t u(t, x) := \frac{\partial}{\partial t} u(t, x), \quad \partial_{tt}^2 u(t, x) := \frac{\partial^2}{\partial t^2} u(t, x)$$

para as primeiras e segundas derivadas parciais de u com respeito a t , e notações semelhantes para derivadas parciais com respeito a x .

A equação (1) se chama *equação da onda*, e modela a propagação de ondas em um meio heterogêneo no intervalo de tempo $[0, T]$. A função u solução da equação da onda representa a *pressão acústica* da onda. As condições (2) e (3) são *condições iniciais* de posição e de velocidade, respectivamente. As funções u_0 e u_1 são dadas, u_0 representa uma posição inicial e u_1 representa uma velocidade inicial. As condições (4)-(5) são *condições de fronteira de Dirichlet*. A função f é uma *fonte* dada. A constante c é a velocidade da onda, suponhamos que ela é conhecida. O problema (1)-(5) é um *problema direto bem-posto*, isto significa que ele tem uma solução única.

2.1 Um método explícito simples

Queremos aproximar numericamente a solução de (1)-(5) para uma fonte f dada. A ideia mais simples para obter uma aproximação numérica das derivadas parciais é de aproximá-las por *diferenças finitas centradas*:

$$\partial_t u(t, x) \approx \frac{u(t + \Delta t/2, x) - u(t - \Delta t/2, x)}{\Delta t} \quad \text{e} \quad \partial_x u(t, x) \approx \frac{u(t, x + \Delta x/2) - u(t, x - \Delta x/2)}{\Delta x},$$

onde Δt e Δx são pequenas variações no tempo e no espaço, respectivamente. Observe que essas diferenças finitas convergem para as derivadas respectivas quando Δt e Δx convergem para 0. Da mesma maneira, consideramos as aproximações seguintes para as derivadas segundas:

$$\partial_{tt}^2 u(t, x) \approx \frac{\partial_t u(t + \Delta t/2, x) - \partial_t u(t - \Delta t/2, x)}{\Delta t} \quad \text{e} \quad \partial_{xx}^2 u(t, x) \approx \frac{\partial_x u(t, x + \Delta x/2) - \partial_x u(t, x - \Delta x/2)}{\Delta x}.$$

Aproximando de novo $\partial_t u$ e $\partial_x u$ nestas expressões, obtemos as aproximações seguintes para as derivadas segundas:

$$\partial_{tt}^2 u(t, x) \approx \frac{u(t + \Delta t, x) - 2u(t, x) + u(t - \Delta t, x)}{\Delta t^2}, \quad (6)$$

$$\partial_{xx}^2 u(t, x) \approx \frac{u(t, x + \Delta x) - 2u(t, x) + u(t, x - \Delta x)}{\Delta x^2}. \quad (7)$$

Agora vamos considerar discretizações $t_i = i\Delta t$, $i = 0, \dots, n_t$, de $[0, T]$, com $\Delta t = T/n_t$, e $x_j = j\Delta x$, $j = 0, \dots, n_x$, de $[0, 1]$, com $\Delta x = 1/n_x$. Chamamos de u_i^j a aproximação numérica de $u(t_i, x_j)$, i.e.

$u_i^j \approx u(t_i, x_j)$. De acordo com as aproximações (6)-(7) das derivadas segundas, aproximamos a equação da onda (1) da seguinte maneira:

$$\frac{u(t_i + \Delta t, x_j) - 2u(t_i, x_j) + u(t_i - \Delta t, x_j)}{\Delta t^2} - c^2 \frac{u(t_i, x_j + \Delta x) - 2u(t_i, x_j) + u(t_i, x_j - \Delta x)}{\Delta x^2} \approx f(t_i, x_j)$$

Usando a aproximação $u_i^j \approx u(t_i, x_j)$, a solução numérica deve satisfazer então

$$\frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{\Delta t^2} - c^2 \frac{u_i^{j+1} - 2u_i^j + u_i^{j-1}}{\Delta x^2} = f_i^j, \quad (8)$$

onde $f_i^j := f(t_i, x_j)$. Reorganizando (8) obtemos

$$u_{i+1}^j = 2u_i^j - u_{i-1}^j + \Delta t^2 \left(f_i^j + c^2 \frac{u_i^{j+1} - 2u_i^j + u_i^{j-1}}{\Delta x^2} \right), \quad \text{para } i \geq 1 \quad (9)$$

ou de maneira equivalente

$$u_{i+1}^j = -u_{i-1}^j + 2(1 - \alpha^2)u_i^j + \alpha^2(u_i^{j+1} + u_i^{j-1}) + \Delta t^2 f_i^j, \quad \text{para } i \geq 1 \quad (10)$$

com $\alpha = c\Delta t/\Delta x$. A equação (10) pode ser usada para atualizar u_{i+1}^j se $u_i^j, u_{i-1}^j, u_i^{j+1}, u_i^{j-1}$ forem conhecidos. Quando u_0^j e u_1^j são dados para todos $j = 0, \dots, n_x$, os u_i^j podem ser calculados iterativamente para todos $i = 0, \dots, n_t$ usando (10). Os u_0^j e u_1^j podem ser deduzidos das condições iniciais de posição e velocidade $u_0(x)$ e $u_1(x)$ aparecendo em(2)-(3). Neste exercício computacional, sempre escolheremos $u_0^j = 0$ e $u_1^j = 0$ para todos $j = 0, \dots, n_x$ (isto corresponde a posição e velocidade iniciais nulas). Dizemos que (10) é um *método explícito* pois u_{i+1}^j depende explicitamente de $u_i^j, u_{i-1}^j, u_i^{j+1}, u_i^{j-1}$ em (10).

Na hora da implementação, é importante não esquecer as condições de Dirichlet (4)-(5). Isso significa que para $j = 0$ e $j = n_x$, não usaremos a formula (10), mas deveremos atribuir os valores $u_{i+1}^0 = 0$ e $u_{i+1}^{n_x} = 0$ para todos $i = 0, \dots, n_t$. Ao plotar a solução, é bom verificar se a solução u_i^j fica fixa nos pontos $j = 0$ e $j = n_x$.

2.2 Exercício 1: resolver a equação da onda

Neste exercício 1 usaremos $T = 1$, e para a fonte escolhemos

$$f(t, x_c) = 1000c^2(1 - 2\beta^2 t^2 \pi^2)e^{-\beta^2 t^2 \pi^2}, \quad \text{para todos } t \in [0, T]$$

e $f(t, x) = 0$ para todos $x \neq x_c$ (dizemos que f é uma fonte pontual), com os parâmetros $\beta = 10$ e $x_c = 0.7$.

- Primeiro, escolhemos $c^2 = 10$ e $\Delta x = 0.01$. Implemente o método explícito (10) para calcular a aproximação numérica u_i^j com $n_t = 350$.
- No seu código, inclua uma função para plotar a função $x \mapsto u(x, \hat{t})$ num instante \hat{t} dado.
- Em seguida, reduza o valor de n_t de 10 em 10, isto é $n_t = 350, 340, 330$, etc ... A aproximação numérica u_i^j quebra de repente a partir de algum valor crítico de n_t , no sentido que os valores de $u(x, t)$ são subitamente extremamente altos ou $u(x, t)$ oscila amplamente. No seus testes numéricos, a partir de qual valor crítico de n_t a aproximação numérica u_i^j não funciona mais?
- Faça o mesmo trabalho que nos itens precedentes, mas agora com $c^2 = 20$, começando com $n_t = 500$ e reduzindo o valor de n_t de 10 em 10. A partir de qual valor de n_t a aproximação numérica u_i^j não funciona mais? Como pode-se explicar que este valor de n_t é maior que no caso $c^2 = 10$?
- Inclua no seu relatório uma página com vários gráficos de $x \mapsto u(x, \hat{t})$ em instantes $\hat{t} \in [0, T]$, com $c^2 = 20$, para valores altos de n_x e n_t a sua escolha. Escolhe pelo menos $n_x = 200$ e $n_t = 1500$. Por exemplo, podem ser 6 gráficos nos instantes $\hat{t} = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6$ (ver Figura 2 para um exemplo com 2 gráficos).

Esses resultados mostram que usando um método explícito, o passo Δt precisa ser suficientemente pequeno, ou então o método numérico ficará instável. A condição exata sobre Δt para obter estabilidade depende de c e Δx . Esta condição de estabilidade chama-se condição CFL (Courant-Friedrichs-Lax).

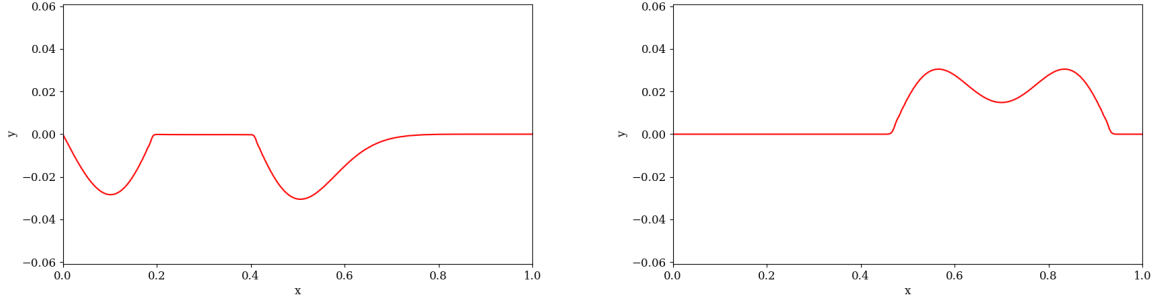


Figura 2: Solução $x \mapsto u(x, \hat{t})$ em $\hat{t} = 0.2$ (esquerda) e $\hat{t} = 0.5$ (direita).

3 Descrição do problema inverso

Agora consideramos o problema seguinte (a partir de agora e até o fim deste EP, para simplificar vamos escolher as condições iniciais $u_0(x) = 0$ e $u_1(x) = 0$ para todos $x \in [0, 1]$)

$$\partial_{tt}^2 u^*(t, x) - c^2 \partial_{xx}^2 u^*(t, x) = \sum_{k=1}^K a_k^* f_k(t, x) \text{ em } [0, T] \times [0, 1], \quad (11)$$

$$u^*(0, x) = 0 \text{ em } [0, 1], \quad (12)$$

$$\partial_t u^*(0, x) = 0 \text{ em } [0, 1], \quad (13)$$

$$u^*(t, 0) = 0 \text{ em } [0, T], \quad (14)$$

$$u^*(t, 1) = 0 \text{ em } [0, T]. \quad (15)$$

Neste problema, as funções f_k , $k = 1, \dots, K$, são conhecidas, mas os coeficientes $a_k^* \in \mathbb{R}$ são incógnitos. Chamaremos os f_k de *fontes* e os a_k^* de *intensidades*.

O problema inverso consiste em reconstruir as intensidades $a_k^* \in \mathbb{R}$, $k = 1, \dots, K$, a partir de medições de u^* em apenas um ponto $x_r \in [0, 1]$. Suponhamos que temos um aparelho capaz de medir $u^*(x_r, t)$ em algum ponto x_r de $[0, 1]$ para todos os tempos $t \in [t_i, t_f]$, onde $0 \leq t_i < t_f \leq 1$. Este tipo de aparelho chama-se frequentemente de *receptor*. Vamos usar a notação $d_r(t) = u^*(x_r, t)$, $t \in [t_i, t_f]$ para essas medições, e chamar elas de *sismograma* (ver Figura 3 para exemplos de sismograma). Vamos supor as medições $d_r(t)$, $t \in [t_i, t_f]$, ficam a disposição, mas que u^* não é conhecido. Queremos reconstruir as intensidades a_k^* .

Como as $a_k^* \in \mathbb{R}$ são incógnitas, mas as fontes f_k , $k = 1, \dots, K$, são conhecidas, vamos definir um outro problema

$$\partial_{tt}^2 u(t, x) - c^2 \partial_{xx}^2 u(t, x) = \sum_{k=1}^K a_k f_k(t, x) \text{ em } [0, T] \times [0, 1], \quad (16)$$

$$u(0, x) = 0 \text{ em } [0, 1], \quad (17)$$

$$\partial_t u(0, x) = 0 \text{ em } [0, 1], \quad (18)$$

$$u(t, 0) = 0 \text{ em } [0, T], \quad (19)$$

$$u(t, 1) = 0 \text{ em } [0, T]. \quad (20)$$

onde os a_k são parâmetros. Agora vamos definir uma *função custo*

$$E(a_1, \dots, a_K) = \frac{1}{2(t_f - t_i)} \int_{t_i}^{t_f} (u(x_r, t) - d_r(t))^2 dt.$$

Observamos que $E(a_1, \dots, a_K) \geq 0$ e que

$$E(a_1^*, \dots, a_K^*) = \frac{1}{2(t_f - t_i)} \int_{t_i}^{t_f} (u^*(t, x_r) - d_r(t))^2 dt = 0.$$

Esse mostra que uma solução do problema

$$\underset{(a_1, \dots, a_K) \in \mathbb{R}^K}{\text{minimize}} E(a_1, \dots, a_K) \quad (21)$$

é de fato $(a_1, \dots, a_K) = (a_1^*, \dots, a_K^*)$. O problema (21) é um problema de mínimos quadrados para os parâmetros (a_1, \dots, a_K) . Para enxergar melhor a dependência de $E(a_1, \dots, a_K)$ nos parâmetros (a_1, \dots, a_K) , vamos introduzir as funções auxiliares u_k soluções de

$$\partial_{tt}^2 u_k(t, x) - c^2 \partial_{xx}^2 u_k(t, x) = f_k(t, x) \text{ em } [0, T] \times [0, 1], \quad (22)$$

$$u_k(0, x) = 0 \text{ em } [0, 1], \quad (23)$$

$$\partial_t u_k(0, x) = 0 \text{ em } [0, 1], \quad (24)$$

$$u_k(t, 0) = 0 \text{ em } [0, T], \quad (25)$$

$$u_k(t, 1) = 0 \text{ em } [0, T]. \quad (26)$$

Observe que $u_k(t, x)$ não depende dos a_k . Como o sistema de equações (16)-(20) é linear, temos claramente $u(t, x) = \sum_{k=1}^K a_k u_k(t, x)$. Assim, obtemos

$$E(a_1, \dots, a_K) = \frac{1}{2(t_f - t_i)} \int_{t_i}^{t_f} \left(\sum_{k=1}^K a_k u_k(t, x_r) - d_r(t) \right)^2 dt. \quad (27)$$

Na forma (27), a função custo E pode ser estudada usando as ferramentas de MMQ desenvolvidas em sala de aula.

3.1 Exercício 2: resolução do problema inverso

- Escreva o sistema normal na forma matricial $Ba = c$ para o problema dos mínimos quadrados (21), usando (27) (escreva os detalhes do cálculo que leva a $Ba = c$). Aqui $a = (a_1, \dots, a_K)$ é o vetor das intensidades a reconstruir, $B \in \mathbb{R}^{K, K}$ é uma matriz quadrada simétrica, e $c = (c_1, \dots, c_K)$ é um vetor.
- Escreva um código que monta o sistema normal $Ba = c$ para um K geral. Observe que B é uma matriz simétrica, portanto, não é necessário calcular todos os coeficientes de B . Para aproximar integrais numericamente, vamos usar a fórmula dos trapézios seguinte:

$$\int_{s_0}^{s_n} g(s) ds \approx \frac{\Delta t}{2} \left[g(s_0) + g(s_n) + 2 \sum_{i=1}^{n-1} g(s_i) \right]$$

onde $s_i = s_0 + i\Delta t$, $i = 0, \dots, n$, $n = (s_n - s_0)/\Delta t$ (escolhe s_n e s_0 de maneira que n seja um inteiro). Observe que a definição de Δt é $\Delta t = T/n_t$, confira Seção 2.1.

- Teste seu código com os dados seguintes: $K = 3$,

$$f_k(t, x_k) = 1000c^2(1 - 2\beta^2 t^2 \pi^2) e^{-\beta^2 t^2 \pi^2}, \quad \text{para todos } t \in [0, T] \quad (28)$$

e $f_k(t, x) = 0$ para todos $x \neq x_k$. Escolhemos os parâmetros $\beta = 10$, $c^2 = 20$, $x_r = 0.7$, $\Delta x = 0.01$, $n_t = 1000$ e $(x_1, x_2, x_3) = (0.2, 0.3, 0.9)$, $t_i = 0.5$ e $t_f = T = 1$. O sismograma d_r^3 é dado (ver arquivo `dr3.npy` ou `dr3.txt`). Observação: o arquivo d_r^3 corresponde a função $d_r(t)$ no intervalo $[0, T]$, isto é d_r^3 tem $n_t + 1 = 1001$ linhas. Para calcular as integrais aparecendo no sistema normal, precisa extrair os valores de d_r^3 no intervalo $[t_i, t_f]$.

- Escreva um código que resolva o sistema normal $Ba = c$ para um K geral usando uma fatoração de Cholesky. A fatoração de Cholesky de B é da forma $B = LL^T$ onde L é uma matriz triangular inferior com entradas diagonais positivas e reais, e L^T denota a matriz transposta de L .
- Resolva o sistema normal $Ba = c$ no caso $K = 3$ descrito no item anterior usando seu código de fatoração de Cholesky. Neste caso os parâmetros ótimos $(a_1^*, a_2^*, a_3^*) = (0.1, 40.0, 7.5)$ são conhecidos. Seu algoritmo deveria fornecer estes parâmetros ótimos com uma precisão alta. Vamos denotar $(\tilde{a}_1, \tilde{a}_2, \tilde{a}_3)$ a solução numérica obtida com seu algoritmo. Calcule numericamente o erro de reconstrução seguinte:

$$\text{erro}_{L2} = \sqrt{\sum_{k=1}^K (\tilde{a}_k - a_k^*)^2}.$$

Calcule também o resíduo :

$$\text{resíduo} = E(\tilde{a}_1, \dots, \tilde{a}_K).$$

Observe que ambos o erro $_{L2}$ e o resíduo devem ser pequenos, pois os dois estão medindo a distancia da reconstrução $(\tilde{a}_1, \dots, \tilde{a}_K)$ à solução exata (a_1^*, \dots, a_K^*) .

- Escreva um código que resolva o sistema normal $Ba = c$ para um K geral usando um método iterativo SOR com $\omega = 1.6$ e com 10000 iterações. Calcule o erro $_{L2}$ e o resíduo. Qual do método SOR ou da fatoração de Cholesky fornece o menor erro $_{L2}$ e o menor resíduo?
- Agora faça os mesmos passos no caso seguinte: $K = 10$, e as funções f_k são fontes pontuais definidas em (28) com

$$(x_1, \dots, x_K) = (0.03, 0.15, 0.17, 0.25, 0.33, 0.34, 0.40, 0.44, 0.51, 0.73).$$

Os parâmetros são $\beta = 10$, $c^2 = 20$, $x_r = 0.7$, $\Delta x = 0.01$, $n_t = 1000$, $t_i = 0.9$ e $t_f = T = 1$. Nesse exercício, os parâmetros ótimos são conhecidos, iguais a

$$(a_1^*, \dots, a_K^*) = (7.3, 2.4, 5.7, 4.7, 0.1, 20.0, 5.1, 6.1, 2.8, 15.3).$$

O sismograma d_r^{10} é dado (ver arquivo `dr10.npy` ou `dr10.txt`. Observação: o arquivo d_r^{10} corresponde a função $d_r(t)$ no intervalo $[0, T]$, isto é d_r^{10} tem $n_t + 1 = 1001$ linhas.)

- Agora vamos considerar um caso com $K = 20$, $\beta = 10$, $c^2 = 20$, $x_r = 0.7$, $\Delta x = 0.005$, $n_t = 2000$, $t_i = 0.9$ e $t_f = T = 1$. As funções f_k são fontes pontuais definidas em (28) com

$$x_k = 0.1 + 0.025(k - 1), \quad k = 1, \dots, 20.$$

A tarefa é de calcular $(\tilde{a}_1, \dots, \tilde{a}_K)$ solução de $Ba = c$ usando seu algoritmo de fatoração de Cholesky. Neste caso d_r^{20} é dado (ver arquivo `dr20.npy` ou `dr20.txt`. Observação: o arquivo d_r^{20} corresponde a função $d_r(t)$ no intervalo $[0, T]$, isto é d_r^{20} tem $n_t + 1 = 2001$ linhas.) mas a solução exata (a_1^*, \dots, a_K^*) não é conhecida. No seu relatório, escreva os valores que você encontra para $(\tilde{a}_1, \dots, \tilde{a}_K)$ na mesma ordem dada pelo seu algoritmo e, além disso, mostre os valores de $(\tilde{a}_1, \dots, \tilde{a}_K)$ classificados em ordem crescente (para ajudar na verificação dos valores).

3.2 Exercício 3: estudo da influência do ruído na reconstrução

No exercício 2, os sismogramas d_r^3 e d_r^{10} eram idealizados, no sentido que eles eram medições exatas de $u^*(t, x_r)$ no intervalo $[t_i, t_f]$. No entanto, na prática os sismogramas sempre tem defeitos, por várias razões: o sinal é sempre poluído por ruído, os aparelhos de medição não são perfeitos, e o modelo matemático é apenas uma aproximação da situação real. No exercício 2, obtivemos reconstruções excelentes das intensidades (a_1^*, \dots, a_{10}^*) , no sentido que o erro $_{L2}$ e o resíduo eram extremamente pequenos. Isso pode dar a impressão de que este problema inverso é fácil de resolver, no entanto, veremos neste exercício que uma quantidade muito pequena de ruído nos dados d_r pode levar a um resultado completamente errado. Isso mostra a instabilidade desse problema inverso, e por isso este problema inverso é chamado de mal-posto.

Neste exercício 3, vamos considerar o mesmo caso que no exercício 2 com $K = 10$, mas agora vamos usar um sismograma ruidoso, que chamaremos de d_r^{ruido} , em vez de d_r^{10} . Vamos definir um sismograma ruidoso a partir de d_r^{10} da seguinte maneira:

$$d_r^{\text{ruido}}(t) := (1 + \delta(t))d_r^{10}(t) \quad \text{para todos } t \in [t_i, t_f],$$

onde

$$\delta(t) = \left(\eta \max_{s \in [t_i, t_f]} |d_r^{10}(s)| \right) v(t),$$

e $v(t)$ tem valores aleatórios entre -1.0 e 1.0 . Por exemplo, pode usar-se a função `numpy.random.rand` para definir $v(t)$. Aqui, η é um parâmetro que usaremos para controlar o nível do ruído.

Vamos definir o *nível de ruído* da seguinte maneira

$$\text{ruído} = 100 \frac{\int_{t_i}^{t_f} |d_r^{10}(t) - d_r^{\text{ruido}}(t)| dt}{\int_{t_i}^{t_f} |d_r^{10}(t)| dt}.$$

Observe que o nível de ruído é uma porcentagem, por exemplo se calculamos ruído = 1.5, isto significa que o nível de ruído nos dados é de 1.5%.

- Implemente no seu código os dados ruidosos d_r^{ruido} e o nível de ruído como definidos acima. Implemente também uma função que plota os dados sem ruído d_r^{10} e os dados ruidosos d_r^{ruido} no intervalo $[t_i, t_f]$ (Coloque os dois plots em uma figura só - ver Figura 3 para um exemplo de tal plotagem).
- Resolva o sistema normal $Ba = c$ usando d_r^{ruido} em vez de d_r^{10} com os valores $\eta = 10^{-3}, 10^{-4}, 10^{-5}$ e calcule o nível de ruído correspondente. Calcule o erro L_2 e o resíduo. Plote também os dados sem ruído d_r^{10} e os dados ruidosos d_r^{ruido} no intervalo $[t_i, t_f]$. Observe que a diferença entre os dados ruidosos e os dados sem ruído é visualmente imperceptível para estes valores de η . Discuta a influência do nível de ruído no erro de reconstrução e nos resíduos. Deve-se observar que, mesmo com uma pequena quantidade de ruído, a reconstrução $(\tilde{a}_1, \dots, \tilde{a}_K)$ é bastante degradada.

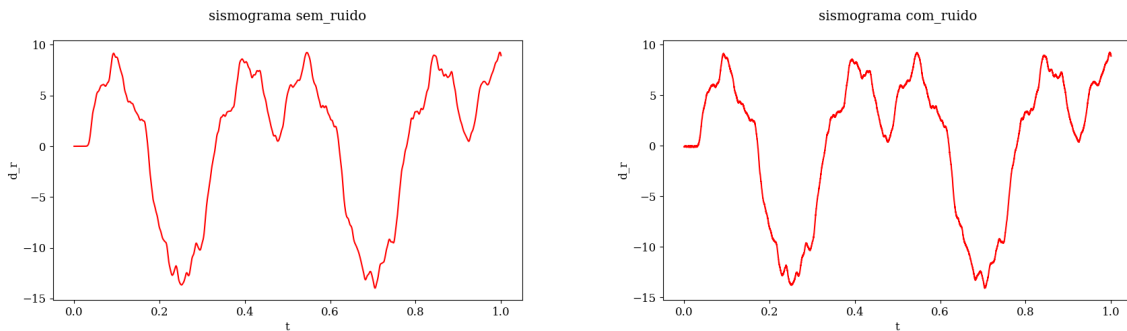


Figura 3: Exemplo de sismograma sem ruído (esquerda) e com 0.96% de ruído (direita)

4 Observações finais

O problema de reconstrução dos parâmetros (a_1^*, \dots, a_K^*) visto nesse EP é uma versão bastante simplificada de um problema inverso altamente relevante da geofísica. Este problema encontra-se frequentemente na literatura com o nome *full waveform inversion*. Ele tem aplicações chaves na indústria e na geofísica, tais como o mapeamento do subsolo para prospecção de petróleo e a detecção da origem de terremotos. O problema de full waveform inversion é um assunto moderno de pesquisa, e atualmente existe uma grande comunidade de engenheiros, geofísicos e matemáticos trabalhando nesse assunto.

Em uma aplicação mais realista, a primeira dificuldade adicional é que o problema é estudado em dimensões 2 ou 3, o que dificulta bastante a implementação numérica da equação da onda, tanto do ponto de vista matemático quanto do computacional. A outra dificuldade adicional é que a dependência da função custo nos parâmetros é muito mais complexa em problemas realistas. Por exemplo, um problema mais realista seria de reconstruir as posições das fontes pontuais f_k em vez de reconstruir apenas a intensidade de f_k . Nesse caso a dependência da função custo já é bem mais complicada, e o sistema normal não é de fácil cálculo. O problema inverso de reconstrução da posição de uma fonte pontual tem aplicação em detecção da origem de terremotos por exemplo. Um outro exemplo altamente relevante é de reconstruir a velocidade c na equação da onda (11) em meios heterogêneos. Nesse caso c é uma função, e o mapeamento dessa função traz muitas informações relevantes sobre o subsolo.

5 Instruções, observações e dicas

- **Esse exercício pode ser feito em duplas.**
- O uso de bibliotecas **não é permitido** (por exemplo para realizar as operações entre matrizes, ou para fazer a fatoração de Cholesky, ou para usar o método iterativo SOR). A única exceção é que funções que permitem a representação de matrizes e vetores são permitidas (por exemplo, a função `numpy.array`).

- O programa deverá ser escrito em Python, usando o pacote `numpy`, ou em linguagem C. O seu código deverá estar bem comentado e estruturado. A entrada e a saída deverão ser feitas de forma a ajudar o usuário a executar o programa e devem facilitar a análise dos resultados. Se o seu programa precisa de arquivos de entrada, considere que os mesmos encontram-se na mesma pasta do executável, ou faça de forma que solicite o caminho/nome do arquivo ao usuário.
- As análises e resultados obtidos devem ser organizados em um relatório que deve minimamente discutir os problemas estudados e os resultados obtidos. A entrega deverá conter um relatório (no formato `.pdf`), contendo a análise do problema estudado, e o código usado para as simulações computacionais (arquivos `.py` ou `.c`, tentando minimizar a quantidade de arquivos). A entrega também deverá ser feita em um arquivo compactado único (por exemplo um arquivo `.zip`).
- O uso de \LaTeX para escrever o relatório é fortemente incentivado. Os relatórios escritos em Latex receberão um bônus de 0.5 pontos.
- Os arquivos `.npy` são dados em formato binário para usar com `numpy`. Você pode carregar eles diretamente usando:

```
from numpy import load
dr3 = load('dr3.npy')
```

O resultado é um vetor do tipo `numpy.array` com $n_t + 1$ entradas. Alternativamente, os mesmos dados são disponíveis em formato `.txt`.

Critérios de Correção

- Exercício 1 (3.8 pts) (Implementação correta da equação da onda, figuras, valores críticos de n_t corretos, interpretação dos resultados)
- Exercício 2 (3.8 pts) (Cálculo analítico do sistema normal correto, calculo numérico do sistema normal $Ba = c$, implementação da formula dos trapézios, implementação do método SOR, implementação da fatoração de Cholesky, resolução do sistema nos casos $K = 3$ e $K = 10$, cálculo do erro e do resíduo, figuras, discussão dos resultados obtidos)
- Exercício 3 (1.0 pts) (Cálculo de $d_r^{\text{ruído}}$, cálculo do nível de ruído, resolução numérica do sistema normal $Ba = c$ com ruído, discussão dos resultados)
- Código bem documentado: comentários, legibilidade. (0.7 pts)
- Qualidade do relatório (relevância dos comentários e apresentação geral). (0.7 pts)
- Uso de \LaTeX (0.5 pts extra)
- Será verificado se o programa entregue roda e produz saídas consistentes com os resultados apresentados no relatório.
- Em caso de atraso de até 48h, -2 pontos. Após isso, o EP não será aceito.

Referências

- [1] J. L. Mueller and S. Siltanen. *Linear and nonlinear inverse problems with practical applications*, volume 10 of *Computational Science & Engineering*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2012.