Universidade Estadual de Campinas

Instituto de Matemática, Estatística e Computação Científica

Dissertação de Mestrado

Reamostragem Uniforme Utilizando a Função SINC

Autora: **Ana Carolina Camargo**
Orientador: **Prof. Dr. Lúcio Tunes dos Santos**

Campinas, SP
Março 2006

Reamostragem Uniforme Utilizando a Função SINC

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por Ana Carolina Camargo e aprovada pela comissão julgadora.

Campinas, 29 de março de 2006.

Prof. Dr. Lúcio Tunes dos Santos
Orientador

Banca Examinadora
Prof. Dr. Lúcio Tunes dos Santos
Prof. Dr. Alvaro Rodolfo de Pierro
Prof. Dr. Rodrigo de Souza Portugal

Dissertação de Mestrado apresentada no Instituto de Matemática, Estatística e Computação Científica, UNICAMP, como requisito parcial para obtenção do Título de Mestre em Matemática Aplicada.

Reamostragem Uniforme Utilizando a Função SINC

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por Ana Carolina Camargo e aprovada pela comissão julgadora.

Campinas, 29 de março de 2006.

Prof. Dr. Lúcio Tunes dos Santos
Orientador

Banca Examinadora
Prof. Dr. Lúcio Tunes dos Santos
Prof. Dr. Alvaro Rodolfo de Pierro
Prof. Dr. Rodrigo de Souza Portugal

Dissertação de Mestrado apresentada no Instituto de Matemática, Estatística e Computação Científica, UNICAMP, como requisito parcial para obtenção do Título de Mestre em Matemática Aplicada.

Dissertação de Mestrado defendida em 29 de março de 2006 e aprovada

Pela Banca Examinadora composta pelos Profs. Drs.

Prof. (a). Dr (a). LUCIO TUNES DOS SANTOS

Prof. (a). Dr (a). ALVARO RODOLFO DE PIERRO

Prof. (a). Dr (a). RODRIGO DE SOUZA PORTUGAL

Resumo

É comum ser preciso reconstruir funções cujas amostras não estão numa grade igualmente espaçada. Isto é devido ao fato que alguns dos algoritmos mais usados requerem amostras em uma grade Cartesiana regular (uniforme). Portanto, é necessário fazer uma reamostragem uniforme, i.e., interpolar as amostras não uniformes em um conjunto de pontos igualmente espaçados. Neste trabalho, primeiro mostramos que o problema de reamostragem pode ser formulado como um problema de resolver um sistema de equações lineares. Uma solução para este sistema pode ser encontrada utilizando a matriz pseudoinversa, um processo que é impraticável para um número grande de variáveis. A partir de características do problema, é possível desenvolver um algoritmo melhor, o qual usa apenas um número limitado de amostras para calcular cada amostra uniforme, transformando o problema original numa seqüência de sistemas lineares com menos variáveis. O resultado final pode ser visto como ótimo e computacionalmente eficiente. Aplicações são apresentadas para demonstrar a eficiência deste método.

Palavras-chave: Reamostragem, Transformada de Fourier, Função Sinc.

Abstract

Its common to be needed to reconstruct functions which samples falls on a nonequally spaced grid. This is due to the fact that some of the most used algorithms require samples in a regular (uniform) Cartesian grid. Therefore, it is necessary to make an uniform resampling, i.e., to interpolate the nonuniform samples in a set of equally spaced points. In this work, it is first shown that the resampling problem can be formulated as a problem of solving a system of linear equations. A solution for this system can be found using the pseudoinverse matrix, a process that is impractical for a large number of variables. From particular characteristics of the problem, it is possible to develop a better algorithm, which only uses a limited number of samples to calculate each uniform sample, transforming the original problem into a sequence of linear systems with less variables. The final result can be viewed as both optimal and computationally efficient. Applications are presented to demonstrate the efficiency of the method.

Keywords: Resampling, Fourier Transform, Sinc Function.

"Desconfie do destino e acredite em você.
Gaste mais horas realizando que sonhando...
Fazendo que planejando... Vivendo que esperando...
Porque, embora quem quase morre esteja vivo,
quem quase vive já morreu."
(Quase - Luiz Fernando Veríssimo)

Agradecimentos

Ao Lúcio, meu orientador, sou grata por me receber e por tudo que me ensinou.

A minha eterna mestra e amiga Zoraide, que sempre me incentivou a seguir em frente, mesmo não estando presente.

A amiga Sueli, pelo carinho, pela atenção e por todas as vezes que me recebeu em sua casa.

A amiga Valeria, pelo apoio, pela dedicação, pelos conselhos, pelas vezes que me acolheu em sua casa e por seu filhinho Mirko, com quem passei momentos muito agradáveis.

Aos colegas do laboratório, que sempre estiveram dispostos a ajudar.

Aos amigos Alessandra, Fernando e Gabi, pelo apoio, carinho, amizade e companhia, por todos os momentos que passamos juntos.

A minha querida amiga Mariana, pela cumplicidade, pelas boas conversas que tivemos, pelas risadas, pelas vezes que fiquei em sua casa e por me dar o presente de ser "tia do coração" de uma bonequinha tão doce e linda como a Maria Clara.

Aos meus pais, Luiz e Renata, pelo amor, pelo apoio e pela dedicação.

As minhas amadas irmãs, Luiza Helena e Maria Júlia, minhas princesinhas, por existirem, por encherem de encanto e alegria todos os meus dias, tornando mais leves as dificuldades.

Ao meu querido Fernando, por estar sempre ao meu lado, pelos momentos lindos que passamos juntos, pela infinita amizade, pela paciência, por compreender todas as vezes que não pude estar presente ou não pude lhe dar muita atenção e, principalmente, por seu amor.

À Fapesp, pelo apoio financeiro.

Sumário

Introdução

A transformada de Fourier é uma ferramenta matemática muito utilizada em vários campos da ciência, em particular, no processamento de sinais, sons e imagens. Como no cálculo da transformada de Fourier de um sinal $s(t)$ é necessário conhecer seus valores para todo $t \in \mathbb{R}$, em alguns casos, não podemos calculá-la, já que os conhecemos em apenas um conjunto discreto de pontos. Assim, para contornar este problema, um recurso muito utilizado é a Transformada de Fourier Discreta (DFT, do Inglês, *"Discrete Fourier Transform"*), que é uma aproximação para a Transformada de Fourier.

Os algoritmos baseados na DFT fornecem melhores resultados quando as amostras estão igualmente espaçadas. Porém, na prática, nem sempre é possível se obter uma amostra deste tipo. Uma maneira de resolver este problema é fazer uma reamostragem uniforme, i.e., interpolar as amostras não uniformes em um conjunto de pontos igualmente espaçados.

Este problema ocorre em numerosos campos da ciência. Em sísmica, muitos algoritmos de processamento de dados requerem que os dados de entrada possuam intervalos de amostragem iguais. Entretanto, na realidade, dificuldades de posicionamento de fontes e receptores podem fazer com que o tamanho destes intervalos variem de lugar para lugar. Se estas variações forem muito grandes, os traços precisam ser reamostrados.

Utilizando o Teorema de Shannon (Briggs and Henson, 1995), o problema de reamostragem uniforme pode ser formulado como um problema de resolver um sistema de equações lineares

$$Ax = b, \tag{1}$$

onde a matriz A é a matriz dos coeficientes da interpolação, e x e b são vetores das amostras uniformes e não uniformes, respectivamente (Rosenfeld, 1998). Neste trabalho estudamos a reamostragem uniforme e exemplos de sua aplicação, e o apresentamos como descreveremos a seguir.

No Capítulo 1, estudamos a transformada de Fourier, suas propriedades e apresentamos exemplos. Também estudamos o Teorema de Shannon, que é baseado na função seno cardinal

(sinc) (Gearhart and Shultz, 1990),

$$\text{sinc}(t) = \frac{\text{sen}\pi t}{\pi t}, \tag{2}$$

e que depois será utilizado na formulação do problema de reamostragem uniforme. Finalizamos este capítulo com a DFT e com um exemplo de sua aplicação.

Dando continuidade ao trabalho, no Capítulo 2, estudamos o problema de reamostragem uniforme e sua formulação no sistema (1), começando com o caso unidimensional e depois extendendo o problema para o caso bidimensional. Para resolver este sistema, primeiramente estudamos a Decomposição em Valores Singulares (SVD, do Inglês, *"Singular Value Decomposition"*), ferramenta que foi utilizada em três algoritmos analisados, denominados algoritmo de Reamostragem Uniforme (URS, do Inglês, *" Uniforme ReSampling"*), de Reamostragem Uniforme por Blocos (BURS, do Inglês, *"Block Uniform ReSampling"*) (Rosenfeld, 1998) e Reamostragem Uniforme por Blocos Regularizada (rBURS, do Inglês, *"Regularized Block Uniform ReSampling"*) (Rosenfeld, 2002). Em seguida foi estudado o método dos Gradientes Conjugados, e aplicado na resolução do sistema (1), a título de comparação.

No Capítulo 3, apresentamos alguns exemplos, no caso unidimensional e no caso bidimensional. Os métodos descritos anteriormente foram comparados, em diferentes formulações, verificando assim as situações em que cada um era mais eficiente.

Concluímos este trabalho com uma síntese e conclusão dos resultados obtidos, que são apresentadas em Conclusão.

Capítulo 1

Transformada de Fourier

A transformada de Fourier é certamente uma das mais importantes ferramentas matemáticas. Desde sua introdução por Jean Fourier (1768–1830) em 1822, é encontrada em diferentes aplicações, sendo atualmente utilizada em ramos da ciência e tecnologia tais como Ótica, Estatística, Probabilidade, Criptografia, Acústica e processamento de sinais, sons e imagens.

A principal característica da transformada de Fourier é que ela expressa uma função em uma base de funções trigonométricas, i.e., como uma soma de senos e cossenos multiplicadas por seus respectivos coeficientes. Quando aplicada a uma imagem no domínio espacial gera uma informação no domínio da freqüência, em que cada ponto representa a quantidade de uma dada freqüência contida no domínio espacial da imagem.

DEFINIÇÃO 1.1 *Seja $f : \mathbb{R} \to \mathbb{R}$. A transformada de Fourier de $f(t)$, $F(\omega)$, é dada por*

$$F(\omega) \equiv \mathcal{F}[f(t)](\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-i\omega t}\, dt, \tag{1.1}$$

desde que a integral exista. Neste caso,

$$f(t) \equiv \mathcal{F}^{-1}[F(\omega)](t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega)\, e^{i\omega t}\, d\omega, \tag{1.2}$$

para todo t. O símbolo $\mathcal{F}^{-1}[F(\omega)](t)$ denota a transformada inversa de Fourier.

TEOREMA 1.1 *Seja $f : \mathbb{R} \to \mathbb{R}$ satisfazendo as condições de Dirichlet: f é contínua nos subintervalos com limites laterais finitos e absolutamente integrável nestes intervalos, i.e.,*

$\int_{-\infty}^{\infty} |f(\ t)|\ dt < \infty$. *Então vale o seguinte par de transformadas:*

$$F(\omega) \equiv \mathcal{F}[f(t)](\omega) = \int_{-\infty}^{\infty} f(t)\ e^{-i\omega t}\ dt, \tag{1.3}$$

e

$$\frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)\ e^{i\omega t}\ d\omega = \frac{1}{2}[f(t^+) + f(t^-)]. \tag{1.4}$$

onde $f(t^{\pm}) = \lim_{h\to 0} f(t \pm h)$.

PROVA: Ver Oppenheim and Schafer (1989). □

1.1 Propriedades

A transformada de Fourier satisfaz as propriedades abaixo relacionadas:

- Linearidade: Para todo $\alpha \in I\!R$,

$$\mathcal{F}[f(t) + \alpha g(t)](\omega) = \mathcal{F}[f(t)](\omega)\ +\ \alpha\mathcal{F}[g(t)](\omega) = F(\omega)\ +\ \alpha G(\omega). \tag{1.5}$$

- Simetria:

$$f(t) = \frac{1}{2\pi}\mathcal{F}[F(\omega)](-t). \tag{1.6}$$

- Escala: Para todo $0 \neq a \in I\!R$,

$$\mathcal{F}[f(at)](\omega) = \frac{1}{|a|}F\left(\frac{\omega}{a}\right). \tag{1.7}$$

- Deslocamento: Para todo $b \in I\!R$,

$$\mathcal{F}[f(t-b)](\omega) = e^{-i\omega b}F(\omega). \tag{1.8}$$

1.2 Exemplos

Para exemplificarmos e ilustrarmos o cálculo da transformada de Fourier, apresentamos a seguir alguns exemplos, basedos em funções especiais.

1.2.1 Função Retangular I

Como primeiro exemplo, calculemos a transformada de Fourier da função retangular, cuja expressão é dada por

$$f(t) = \begin{cases} A, & |t| \leq t_0 \\ 0, & |t| > t_0, \end{cases} \tag{1.9}$$

com $A \in \mathbb{R}$ e $t_0 > 0$. Assim,

$$\begin{aligned} F(\omega) &= \int_{-\infty}^{\infty} f(t)\, e^{-i\omega t} dt = \int_{-t_0}^{t_0} A\, e^{-i\omega t} dt = \frac{A\, e^{-i\omega t}}{-i\omega}\Big|_{-t_0}^{t_0} \\ &= \frac{A}{-i\omega}\left[e^{-i\omega t_0} - e^{i\omega t_0}\right] = \frac{2\,A}{\omega}\left[\frac{e^{i\omega t_0} - e^{-i\omega t_0}}{2i}\right] = \frac{2\,A}{\omega}\,\text{sen}\omega t_0 \\ &= 2\,A\,t_0\,\frac{\text{sen}\omega t_0}{\omega t_0} = 2\,A\,t_0\,\text{sinc}\left(\frac{\omega t_0}{\pi}\right), \end{aligned} \tag{1.10}$$

onde

$$\text{sinc}(x) = \frac{\text{sen}\pi x}{\pi x}, \tag{1.11}$$

e é conhecida como a função *seno cardinal* (ver Figura 1.1). Note que a freqüência mais impor-

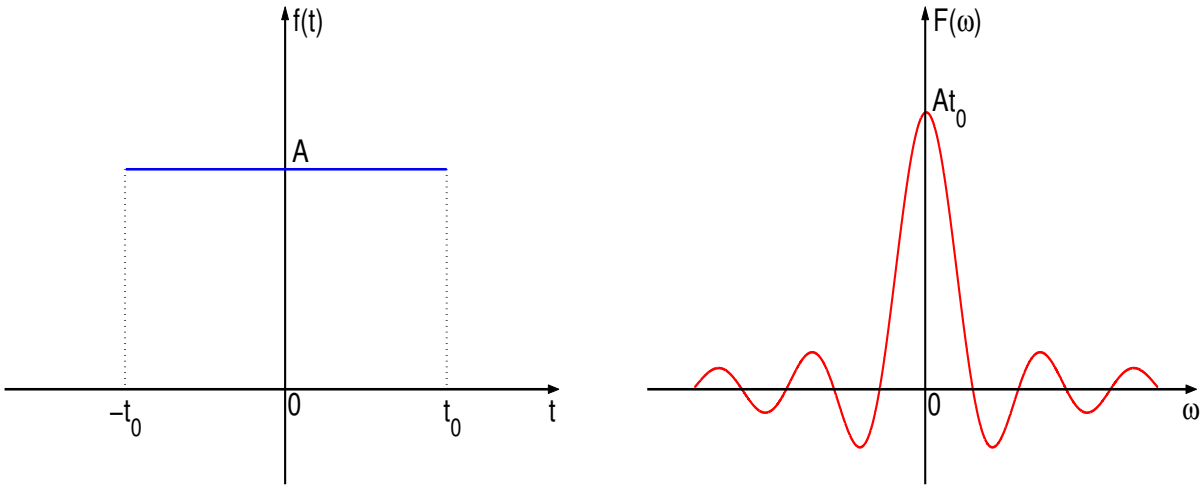


Figura 1.1: Função retangular (à esquerda) e sua transformada de Fourier (à direita).

tante (de maior conteúdo) é a freqüência nula e todas as freqüências estão presentes no sinal, com exceção das freqüências $\omega = k\pi/t_0$, para todo $k \in \mathbb{Z}$. Podemos dizer que este é o "preço" da descontinuidade brusca do sinal. Para maiores detalhes sobre a função sinc veja Gearhart and Shultz (1990).

1.2.2 Função Retangular II

Agora, tomemos uma função retangular no domínio da freqüência, a fim de calcularmos qual $f(t)$ possui esta transformada de Fourier. Assim,

$$F(\omega) = \begin{cases} B, & |\omega| \leq \omega_0 \\ 0, & |\omega| > \omega_0, \end{cases} \tag{1.12}$$

com $B \in I\!R$ e $\omega_0 \in I\!R$.

Aplicando a propriedade da simetria, pelo exemplo anterior temos

$$f(t) = \frac{1}{2\pi} 2\ B\ \omega_0\ \text{sinc}\left(\frac{-\omega_0 t}{\pi}\right) = \frac{B\omega_0}{\pi}\text{sinc}\left(\frac{\omega_0 t}{\pi}\right). \tag{1.13}$$

Concluímos então, que a transformada de Fourier da função sinc é uma função retangular. Note que, $f(t)$ é nula em $t = k\pi/\omega_0$, para todo $k \in \boldsymbol{Z}$ (ver Figura 1.2).

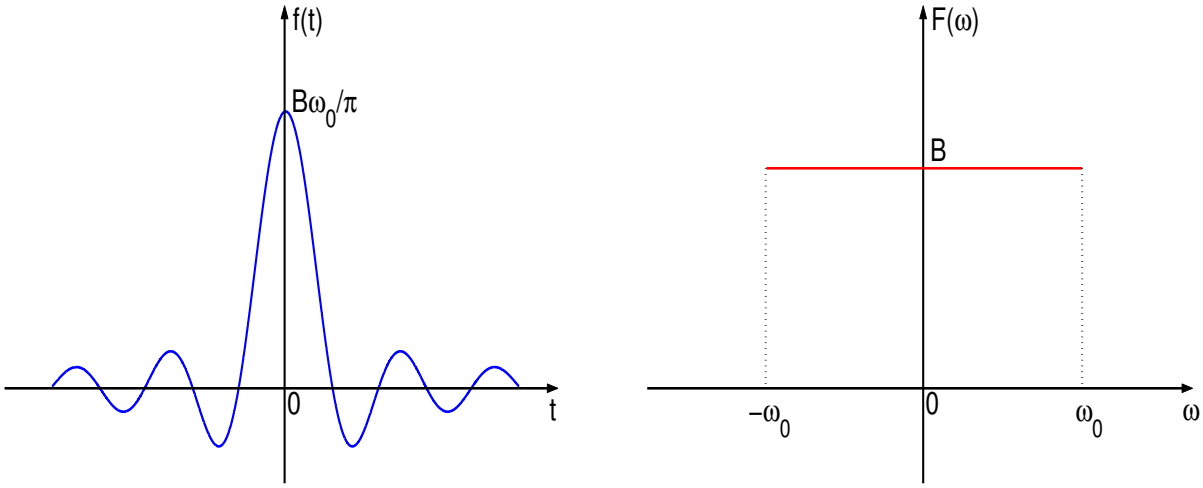


Figura 1.2: Função $f(t) = B\omega_0/\pi\text{sinc}(\omega_0 t/\pi)$ (à esquerda) e sua transformada de Fourier (à direita).

1.2.3 Função Delta de Dirac

A função delta de Dirac, ou função impulso unitário, é geralmente representada por $\delta(t)$ e é muito utilizada em processamento digital de sinais. Uma das definições de tal função, que é baseada em se tomar $A = 1/2\,t_0$ e $t_0 \to 0$ no Exemplo 1.2.1 ($F(\omega) = 1$), é dada por (ver

Figura 1.3)

$$\delta(t) = \mathcal{F}^{-1}[1](t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{i\omega t} d\omega. \tag{1.14}$$

Com base na definição acima apresentada, mostremos uma propriedade importante da função

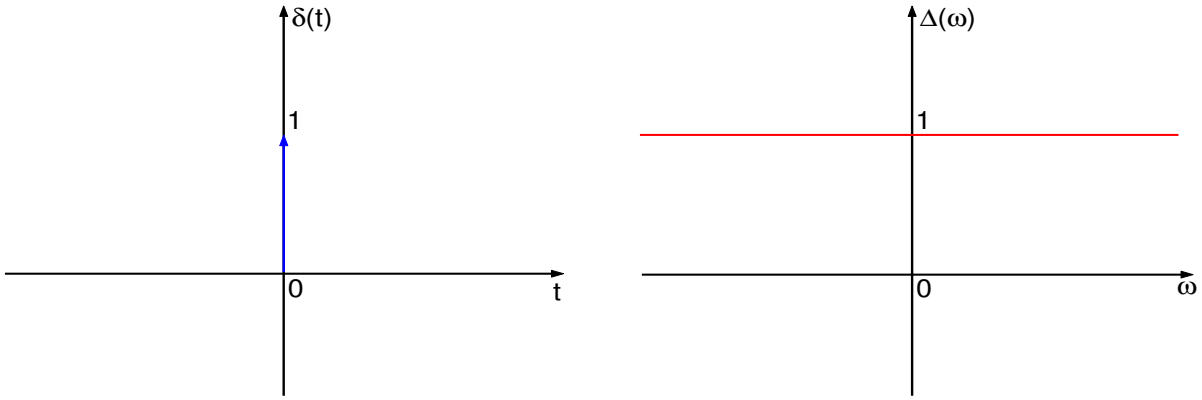


Figura 1.3: Função delta de Dirac (à esquerda) e sua transformada de Fourier (à direita).

delta, que alguns autores apresentam como a própria definição de tal função

$$\begin{aligned}\int_{-\infty}^{\infty} f(t)\,\delta(t)\,dt &= \int_{-\infty}^{\infty}\left[\frac{1}{2\pi}\int_{-\infty}^{\infty} e^{i\omega t} d\omega\right] f(t)\,dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} d\omega \int_{-\infty}^{\infty} f(t)\,e^{i\omega t}\,dt \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} d\omega\; F(-\omega) = -\frac{1}{2\pi}\int_{\infty}^{-\infty} d\omega\; F(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} d\omega\; F(\omega) \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} d\omega\; F(\omega)\,e^{i\omega 0} = f(0),\end{aligned} \tag{1.15}$$

onde $f(t)$ satisfaz algumas condições. Veja Oppenheim and Schafer (1989).

Do resultado acima, segue diretamente

$$\int_{-\infty}^{\infty} f(t)\,\delta(t-a)\,dt = \int_{-\infty}^{\infty} f(t+a)\,\delta(t)\,dt = f(0+a) = f(a). \tag{1.16}$$

Como a função delta é limite de uma função real, ela tem que ser real e portanto temos

$$\delta(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{i\omega t} d\omega = \frac{1}{2\pi}\int_{-\infty}^{\infty} \cos\omega\, t\; d\omega \;+\; i\frac{1}{2\pi}\int_{-\infty}^{\infty} \text{sen}\omega\, t\; d\omega = \frac{1}{\pi}\int_{0}^{\infty} \cos\omega\, t\; d\omega \tag{1.17}$$

e

$$\int_{-\infty}^{\infty} \text{sen}\omega\, t\; d\omega = 0. \tag{1.18}$$

Utilizando o fato de a função cosseno ser par, segue que a função $\delta(t)$ também é uma função par, i.e., $\delta(t) = \delta(-t)$, outra propriedade importante desta função.

1.2.4 Função Cosseno

A transformada de Fourier da função $\cos \omega_0 t$, $\omega_0 \in I\!R$, é dada por

$$\begin{aligned}
\mathcal{F}[\cos \omega_0 t](\omega) &= \int_{-\infty}^{\infty} \cos \omega_0 t \; e^{-i\omega t} \; dt = \int_{-\infty}^{\infty} \left[\frac{e^{i\omega_0 t} + e^{-i\omega_0 t}}{2} \right] e^{-i\omega t} \; dt \\
&= \frac{1}{2} \int_{-\infty}^{\infty} \left[e^{i\omega_0 t - i\omega t} + e^{-i\omega_0 t - i\omega t} \right] dt = \frac{1}{2} \int_{-\infty}^{\infty} e^{it(\omega_0 - \omega)} \; dt \; + \; \frac{1}{2} \int_{-\infty}^{\infty} e^{-it(\omega_0 + \omega)} \; dt \\
&= \pi \; \delta(\omega_0 - \omega) + \pi \; \delta(\omega_0 + \omega) = \pi[\delta(\omega + \omega_0) + \delta(\omega - \omega_0)]. \qquad (1.19)
\end{aligned}$$

Como vemos, a transformada de Fourier é real e é uma soma de funções delta (ver Figura 1.4).

1.2.5 Função Seno

A transformada de Fourier da função $\operatorname{sen} \omega_0 t$, $\omega_0 \in I\!R$, é dada por

$$\begin{aligned}
\mathcal{F}[\operatorname{sen}\omega_0 t](\omega) &= \int_{-\infty}^{\infty} \operatorname{sen}\omega_0 t \; e^{-i\omega t} \; dt = \int_{-\infty}^{\infty} \left[\frac{e^{i\omega_0 t} - e^{-i\omega_0 t}}{2i} \right] e^{-i\omega t} \; dt \\
&= \frac{1}{2i} \int_{-\infty}^{\infty} \left[e^{i\omega_0 t - i\omega t} - e^{-i\omega_0 t - i\omega t} \right] dt = \frac{1}{2i} \int_{-\infty}^{\infty} e^{it(\omega_0 - \omega)} \; dt \; - \; \frac{1}{2i} \int_{-\infty}^{\infty} e^{-it(\omega_0 + \omega)} \; dt \\
&= -i \; \pi \; \delta(\omega_0 - \omega) + i \; \pi \; \delta(\omega_0 + \omega) = i \; \pi[\delta(\omega + \omega_0) - \delta(\omega - \omega_0)]. \qquad (1.20)
\end{aligned}$$

Como vemos, a transformada de Fourier da função seno é imaginária pura e como o cosseno é uma soma de funções delta, (ver Figura 1.4).

1.2.6 Derivada

Um resultado importante é a transformada de Fourier da derivada de uma função $f(t)$, a qual calculamos utilizando integração por partes,

$$\mathcal{F}[f'(t)](\omega) = \int_{-\infty}^{\infty} f'(t) \; e^{-i\omega t} \; dt = \left[e^{-i\omega t} \; f(t) \right] \Big|_{-\infty}^{\infty} - \int_{-\infty}^{\infty} f(t) \; (-i\omega) \; e^{-i\omega t} \; dt = i\omega F(\omega), \qquad (1.21)$$

pois $e^{-i\omega t}$ é limitado, já que ω e t são reais e que $f(t) \to 0$ quando $|t| \to \infty$ pelas hipóteses do Teorema 2.1.

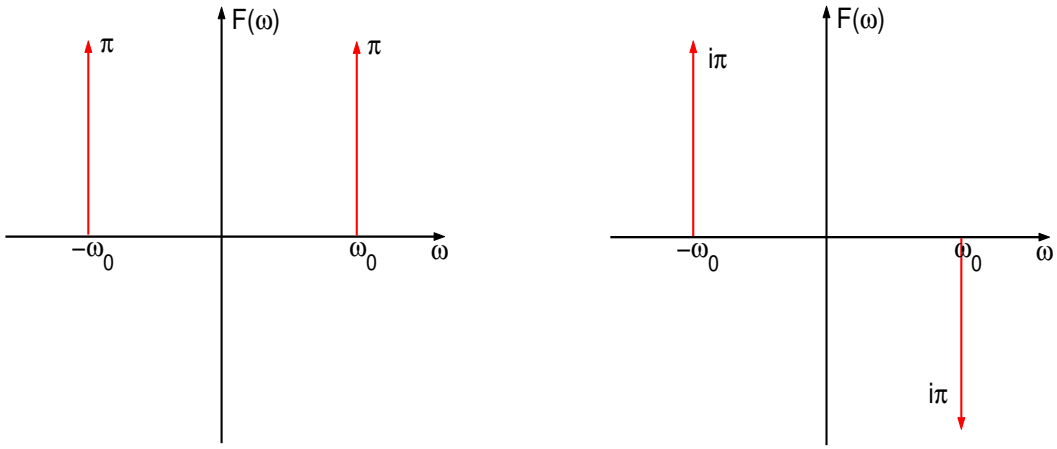


Figura 1.4: Transformada de Fourier da função $\cos \omega_0 t$ (à esquerda) e transformada de Fourier da função $\operatorname{sen} \omega_0 t$ (à direita).

1.2.7 Função Gaussiana

Consideremos a função gaussiana $f \to I\!R : I\!R$, $f(t) = K\ e^{-at^2}$, onde K, $0 \neq a \in I\!R$. Assim,

$$\begin{aligned} F(\omega) &= \int_{-\infty}^{\infty} K\ e^{-at^2}\ e^{-i\omega t}\ dt = K \int_{-\infty}^{\infty} e^{(-at^2 - i\omega t)}\ dt \\ &= K \int_{-\infty}^{\infty} e^{[-(\sqrt{a}\,t\, + \frac{i\omega}{2\sqrt{a}})^2 - \frac{\omega^2}{4a}]}\ dt = K\ e^{-\frac{\omega^2}{4a}} \int_{-\infty}^{\infty} e^{\left[-(\sqrt{a}\,t\, + \frac{i\omega}{2\sqrt{a}})^2\right]}\ dt \\ &= K\ e^{-\frac{\omega^2}{4a}}\ \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} e^{-u^2}\ du = K\sqrt{\frac{\pi}{a}}\ e^{-\frac{\omega^2}{4a}}. \end{aligned} \tag{1.22}$$

Denominando $\bar{K} = K\sqrt{\pi}/(\sqrt{a})$ e $\bar{a} = 1/(4a)$, temos $F(\omega) = \bar{K}\ e^{-\bar{a}\omega^2}$, e, portanto, a transformada de Fourier de uma função gaussiana, também é uma função gaussiana.

1.2.8 Função Degrau Unitário

A função degrau unitário, também conhecida como função de Heaviside, é definida por (ver Figura 1.5)

$$u(t) = \int_{-\infty}^{t} \delta(\tau)\, d\tau = \begin{cases} 1, & t > 0 \\ 1/2, & t = 0 \\ 0, & t < 0, \end{cases} \tag{1.23}$$

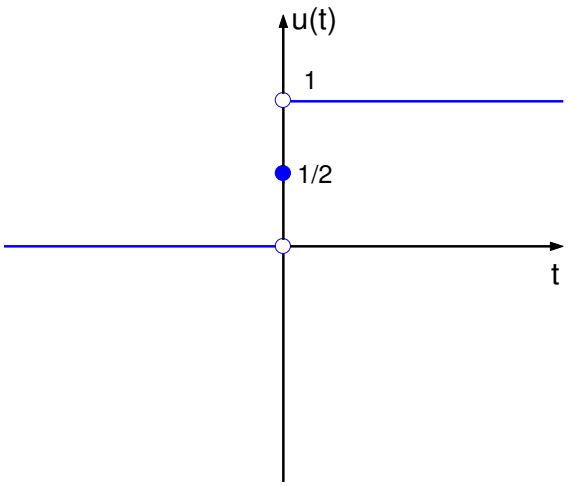


Figura 1.5: Função degrau unitário.

e, de acordo com esta definição, podemos "interpretar" a função delta como a derivada da função degrau unitário, ou seja

$$\delta(t) = \frac{d}{dt}u(t). \tag{1.24}$$

Baseados nos fatos acima relacionados, calculemos a transformada de Fourier de $u(t)$. Consideremos a seguinte relação

$$u(t) + u(-t) = 1, \tag{1.25}$$

válida para todo $t \in I\!R$. Aplicando a transformada de Fourier na equação acima e utilizando as propriedades de Linearidade e de Escala, temos

$$U(\omega) + U(-\omega) = \int_{-\infty}^{\infty} e^{-i\omega t}\, dt = 2\pi\ \delta(\omega). \tag{1.26}$$

Assim, com base na equação anterior, podemos supor que $U(\omega)$ é da forma

$$U(\omega) = \pi\ \delta(\omega) + B(\omega), \tag{1.27}$$

onde B é uma função ímpar. Passando agora a transformada de Fourier na equação (1.24) e utilizando o fato da transformada de Fourier da função delta ser igual a 1, temos

$$i\omega U(\omega) = 1 \tag{1.28}$$

e, substituindo a equação (1.27) na equação acima ,

$$i\omega\ [\pi\ \delta(\omega) + B(\omega)] = 1. \tag{1.29}$$

Percebendo que $\omega\ \delta(\omega) = 0$, podemos escrever

$$B(\omega) = \frac{1}{i\omega}. \tag{1.30}$$

Substituindo agora a equação anterior na equação (1.27) obtemos a transformada de Fourier da função degrau (ver Figura 1.6), dada por

$$U(\omega) = \pi\ \delta(\omega) + \frac{1}{i\omega}. \tag{1.31}$$

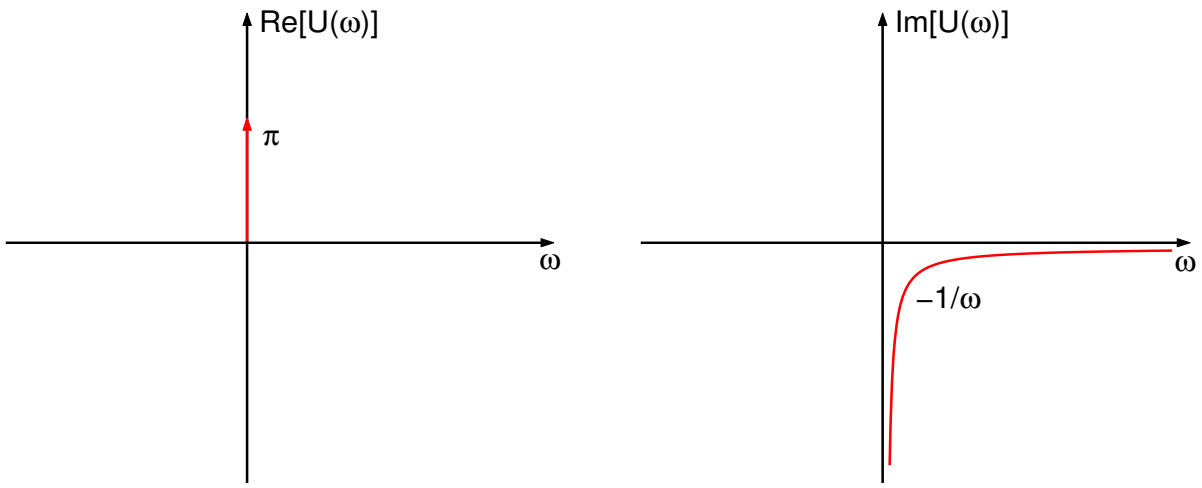


Figura 1.6: Transformada de Fourier da função degrau unitário: parte real (à esquerda), parte imaginária (à direita).

1.3 Teorema da Convolução

Dadas duas funções f e g, a convolução $f * g$ é dada por

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau. \tag{1.32}$$

Com base nesta definição, enunciamos a seguir um teorema conhecido como "Teorema da Convolução", que é um dos resultados mais importantes envolvendo a transformada de Fourier.

TEOREMA 1.2 *Sejam $f(t)$ e $g(t)$ duas funções e $F(\omega)$ e $G(\omega)$ suas respectivas transformadas de Fourier. Então*

$$\mathcal{F}[f(t) * g(t)]\,(\omega) = F(\omega)\,G(\omega). \tag{1.33}$$

PROVA: Usando a definição de convolução temos

$$\begin{aligned}
\mathcal{F}[f(t) * g(t)](\omega) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\tau)\,g(t-\tau)d\tau\; e^{-i\omega t}dt \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} [f(\tau)\,e^{-i\omega\tau}\,d\tau][g(t-\tau)e^{-i\omega(t-\tau)}d\tau] \\
&= \int_{-\infty}^{\infty} f(\tau)\; e^{-i\omega\tau}\; d\tau \int_{-\infty}^{\infty} g(t-\tau)\; e^{-i\omega(t-\tau)}\; d\tau \\
&= F(\omega)\,G(\omega).
\end{aligned} \tag{1.34}$$

Concluímos assim a prova do teorema. □

Pelo teorema da convolução, temos que a convolução no domínio do tempo ou espaço corresponde à multiplicação no domínio da freqüência ou número de onda, e vice-versa. Portanto, se precisarmos calcular a convolução entre duas funções, podemos substituir a convolução pela multiplicação no domínio da freqüência e depois calculamos a transformada de Fourier inversa, obtendo o resultado desejado, com cálculos mais simples.

1.4 Teorema de Shannon

Apresentamos agora um resultado muito importante para o tratamento de sinais digitais, que é o teorema de Shannon, ou, teorema da amostragem, como também é conhecido. Ele estabelece que uma função pode ser completamente determinada a partir de uma amostra discreta, sob algumas hipóteses que seguem.

TEOREMA 1.3 *Seja $f : \mathbb{R} \to \mathbb{R}$ uma função banda limitada, i.e., $F(\omega) = 0$ se $|\omega| > \Omega$ para algum $0 < \Omega < \infty$. Se $\Delta t < \pi/\Omega$, então para todo $t_0 \in \mathbb{R}$,*

$$f(t) = \sum_{j\in\mathbb{Z}} f(t_0 + j\Delta t)\ \text{sinc}\left(\frac{t - t_0 - j\Delta t}{\Delta t}\right). \tag{1.35}$$

PROVA: Expandindo a função $e^{i\omega t}$ em sua série de Fourier (Briggs and Henson, 1995), no intervalo $[-\Omega\ ,\ \Omega]$, temos

$$e^{i\omega t} \ = \ \sum_{n=-\infty}^{\infty} c_n \ e^{-2i\pi n\omega/\Omega}, \tag{1.36}$$

onde

$$c_n = \frac{1}{2\Omega}\int_{-\Omega}^{\Omega} e^{i\omega t}\ e^{-i\pi n\omega/\Omega}\ d\omega. \tag{1.37}$$

Denominando $\Delta t = \pi/\Omega$ e $t_n = n\Delta t$, temos que

$$c_n = \frac{1}{2\Omega}\int_{-\Omega}^{\Omega} e^{i\omega(t-t_n)}\ d\omega = \frac{\text{sen}(\pi(t-t_n)/\Delta t)}{\pi(t-t_n)/\Delta t} = \text{sinc}\left(\frac{t-t_n}{\Delta t}\right), \tag{1.38}$$

e, portanto,

$$e^{i\omega t} = \sum_{n=-\infty}^{\infty} \text{sinc}\left(\frac{t-t_n}{\Delta t}\right) e^{i\omega t_n}, \quad \omega \in [-\Omega,\ \Omega]. \tag{1.39}$$

Combinando este resultado com o fato de $f(t)$ ser banda limitada, segue que

$$\begin{aligned} f(t) \ &= \ \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\ \omega)\ e^{i\omega t}\ d\omega = \frac{1}{2\pi}\int_{-\Omega}^{\Omega} F(\ \omega\ ) \sum_{n=-\infty}^{\infty} \text{sinc}\left(\frac{t-t_n}{\Delta t}\right)\ e^{i\omega t_n} d\omega \\ &= \ \sum_{n=-\infty}^{\infty} \text{sinc}\left(\frac{t-t_n}{\Delta t}\right) \frac{1}{2\pi}\int_{-\Omega}^{\Omega} F(\ \omega\ )\ e^{i\omega t_n} d\omega \\ &= \ \sum_{n=-\infty}^{\infty} \text{sinc}\left(\frac{t-t_n}{\Delta t}\right)\ f(t_n). \end{aligned} \tag{1.40}$$

Concluímos assim a prova do teorema. □

Esse teorema é muito importante, uma vez que fornece o valor exato da função em qualquer ponto, a partir de uma amostra discreta da mesma. Sabemos que na prática, não temos infinitas amostras de uma função, mas se possuirmos uma quantidade considerável de amostras igualmente espaçadas, que satisfaçam as hipóteses do teorema, o resultado ainda é muito bom. A desigualdade $\Delta t < \pi/\Omega$ é conhecida como critério de Nyquist, e muito utilizada no processamento de imagens.

1.5 Transformada de Fourier Discreta

A Transformada de Fourier Discreta (DFT, do Inglês, *"Discrete Fourier Transform"*) pode ser vista como uma aproximação para a transformada de Fourier de uma função, a qual definimos anteriormente. Esta aproximação faz-se necessária pois, na prática, muitas vezes temos que calcular a transformada de Fourier de uma função cujos valores conhecemos apenas em um conjunto discreto de pontos.

A DFT aproxima o cálculo da integral de Riemann da transformada de Fourier, substituindo a integral por uma soma, através da discretização ao longo dos eixos t e ω. Isto é feito da seguinte forma: dada uma amostra de uma função real f, que satisfaz as condicões de Dirichlet, nos seguintes pontos

$$t_j = t_0 + j\Delta t, \quad t_0 \in I\!R, \quad j = 0, 1, \ldots, N-1, \quad \Delta t \in I\!R, \tag{1.41}$$

podemos aproximar a transformada de Fourier de $f(t)$ por

$$\int_{-\infty}^{\infty} f(t)\, e^{-i\omega t}\, dt \approx \Delta t \sum_{j=0}^{N-1} f(t_j)\, e^{-i\omega t_j}. \tag{1.42}$$

Agora, para definirmos uma aproximação para a transformada de Fourier inversa, precisamos de uma discretização do eixo ω. Neste caso, é necessário que tenhamos o cuidado de sermos coerentes com a nossa amostra, pois tal equação tem que fornecer valores exatos ao calcularmos os t_j's iniciais. Para isso, baseados no Teorema de Shannon, procedemos da seguinte maneira: como Δt é dado, definimos $\omega_c = \pi/\Delta t$ e partindo da idéia que precisamos ter pelo menos o mesmo número de pontos para recuperarmos a amostra inicial, definimos

$$\omega_k = k\Delta\omega, \quad k = 0, 1, \ldots, N-1, \quad \Delta\omega = \frac{2\omega_c}{N} = \frac{2\pi}{N\Delta t}. \tag{1.43}$$

E assim, uma aproximação para a transformada de Fourier inversa é dada por

$$\frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)\, e^{i\omega t}\, d\omega \approx \frac{\Delta\omega}{2\pi}\sum_{k=0}^{N-1} F(\omega_k)\, e^{i\omega_k t} = \frac{1}{N\Delta t}\sum_{k=0}^{N-1} F(\omega_k)\, e^{i\omega_k t}. \tag{1.44}$$

As unidades e dimensões estão de acordo com as equações nas formas contínuas. As propriedades básicas que foram apresentadas para a transformada de Fourier contínua, como a linearidade e a simetria, também valem para as aproximações acima.

Na prática, os algoritmos para o cálculo destas aproximações, não usam os fatores externos

Δt e $1/\Delta t$, e precisam ser aplicados posteriormente. Por isso, utiliza-se a seguinte notação, a qual chamamos de DFT,

$$\tilde{F}(\omega_k) = \tilde{F}_k = \sum_{j=0}^{N-1} f_j \; W_N^{-jk}, \tag{1.45}$$

e

$$f(t_j) = f_j = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{F}_k \; W_N^{jk}, \tag{1.46}$$

onde $W_N = e^{i2\pi/N}$ e substituímos t_j e ω_k por $j\Delta t$ e $k\pi/\Delta t$, respectivamente. Verifiquemos que na equação (1.46), realmente vale a igualdade. Temos

$$\tilde{F}_k \; W_N^{jk} = W_N^{jk} \sum_{p=0}^{N-1} f_p \; W_N^{-pk} = \sum_{p=0}^{N-1} f_p \; W_N^{(j-p)k}, \quad \text{para todo } j,\; k \in 0, 1, \ldots, N-1. \tag{1.47}$$

Assim, somando em k dos dois lados da igualdade, temos

$$\sum_{k=0}^{N-1} \tilde{F}_k \; W_N^{jk} = \sum_{k=0}^{N-1} \sum_{p=0}^{N-1} f_p \; W_N^{(j-p)k} \tag{1.48}$$

$$= \sum_{p=0}^{N-1} f_p \sum_{k=0}^{N-1} W_N^{(j-p)k}, \tag{1.49}$$

e, usando o fato que

$$\sum_{k=0}^{N-1} W_N^{(j-p)k} = \begin{cases} 0, & j \neq p \\ N, & j = p, \end{cases} \tag{1.50}$$

temos

$$\sum_{k=0}^{N-1} \tilde{F}_k \; W_N^{jk} = N \; f_j, \tag{1.51}$$

e, portanto, a equação (1.46) é satisfeita.

Note que a DFT existe para todo inteiro k, independente da existência ou não da transformada de Fourier contínua. Note também que esta definição trata a função como banda limitada, ou seja, como se a maior freqüência contida em f fosse $\pi/\Delta t$. Porém, isto nem sempre é válido nas aplicações da DFT, o que acarreta a baixa resolução do método nestes casos. Por outro lado, existem casos em que o resultado é muito bom, mesmo a função não sendo banda limitada. Isto porque a transformada de Fourier de algumas funções tendem a zero, quando ω tende à infinito, como por exemplo, as gaussianas. Note também que além de ser banda limitada, o espaçamento da amostra precisa satisfazer o critério de Nyquist, caso contrário o resultado do

método também não será satisfatório.

As aproximações consideradas são baseadas em amostras uniformes, igualmente espaçadas. Existem também algoritmos que aproximam a transformada de Fourier para amostras não uniformes, porém o resultados não costumam ser tão bons. Para maiores detalhes sobre a DFT, veja Briggs and Henson (1995) e Leite (1998).

1.5.1 Exemplos

Com o objetivo de comprovar a eficiência da DFT, formulamos um exemplo baseado na função $f(t) = e^{-\pi t^2}$. Como $f(t)$ é uma função gaussiana, sua transformada de Fourier também é uma função gaussiana e, portanto, tem valores muito próximos de zero a partir de um certo valor, ao qual chamaremos ω_c. De acordo com o que vimos anteriormente, temos que $F(\omega) = e^{-\omega^2/(4\pi)}$ e portanto $\omega_c = 10$ é uma boa estimativa (ver Figura 1.7). Para que a DFT

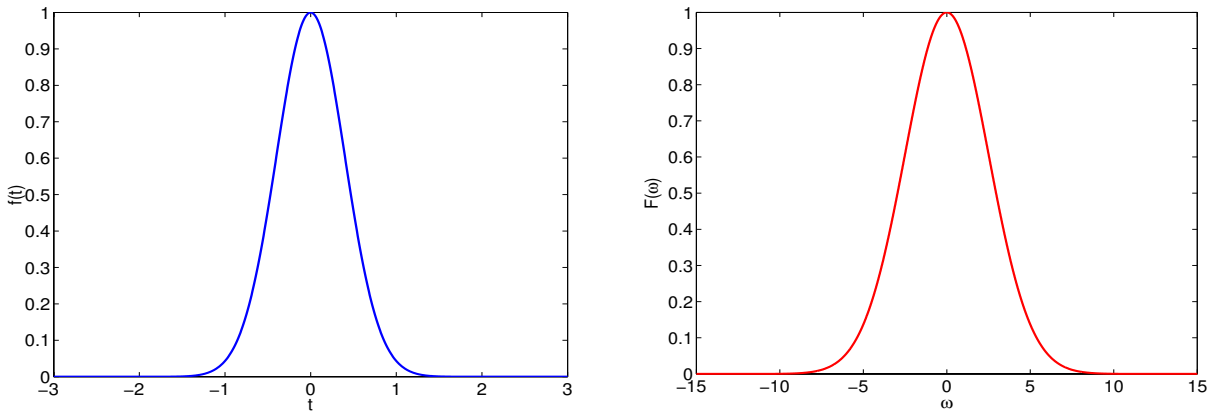


Figura 1.7: Função $f(t) = e^{-\pi t^2}$ (à esquerda) e sua transformada de Fourier $F(\omega) = e^{-\omega^2/(4\pi)}$ (à direita).

forneça bons resultados, a amostra deve satisfazer o critério de Nyquist, ou seja, devemos ter $\Delta t < \pi/\omega_c$, onde Δt é o espaçamento entre as amostras. Como no nosso caso $\omega_c = 10$, devemos ter $\Delta t < 0.31$. Tomamos três amostras igualmente espaçadas de $f(t)$ no intervalo $[-3, 3]$. A primeira com 8 valores ($n = 8$, $\Delta t \approx 0.86$), a segunda com 16 ($n = 16$, $\Delta t = 0.4$) valores e a última com 32 valores ($n = 32$, $\Delta t \approx 0.19$). Conforme o esperado, a DFT da terceira amostra apresentou melhores resultados, já que $\Delta t < 0.31$, aproximando muito bem $F(\omega)$. No caso em que $n = 16$, como Δt estava próximo do desejado, a DFT também apresentou resultados satisfatórios, embora eles não tenham sido tão bons como no terceiro caso. Já no primeiro caso, a DFT apresentou resultados piores que nos demais, pois o espaçamento das amostras era muito maior que o desejado. Os resultados podem ser vistos na Figura 1.8.

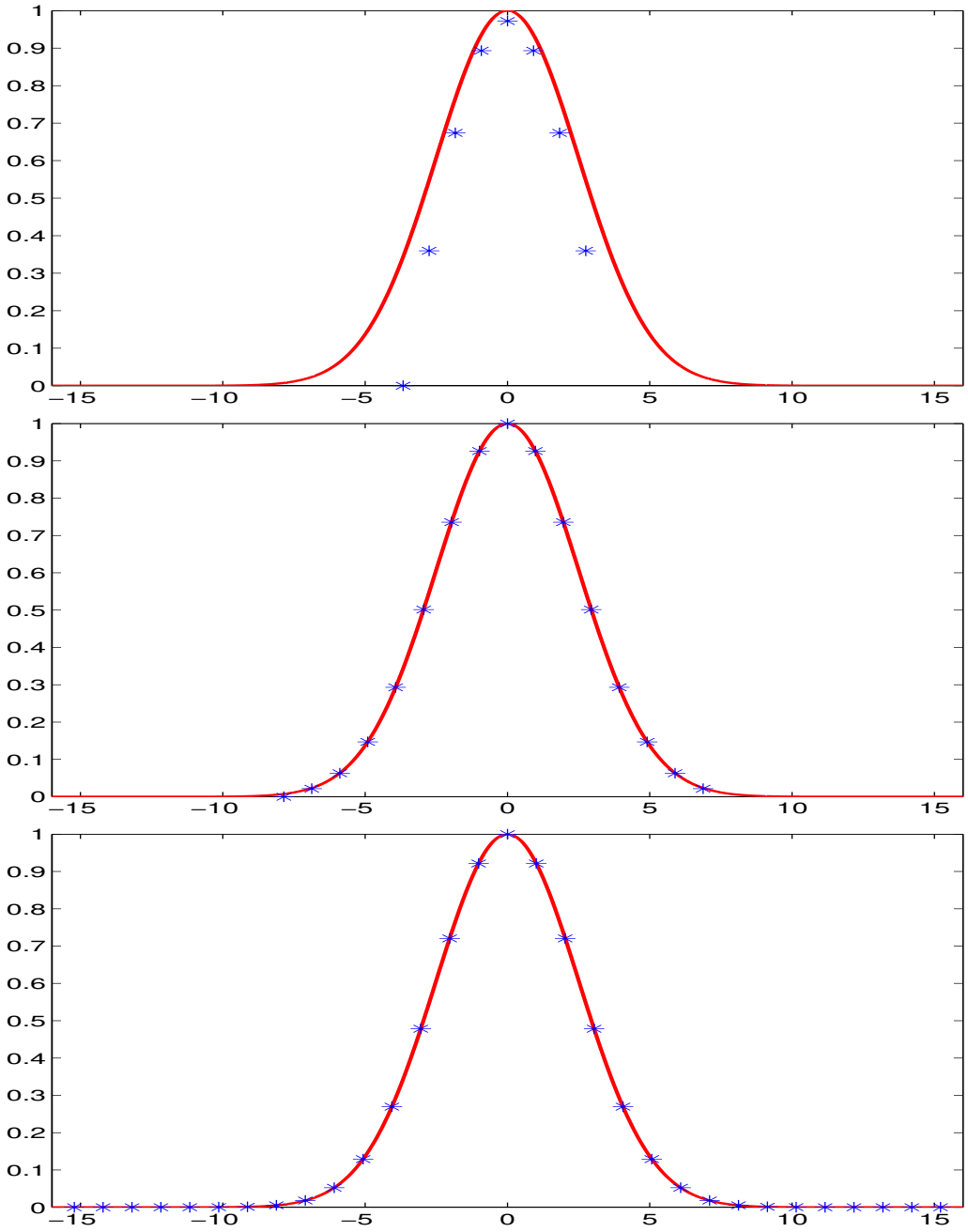


Figura 1.8: DFT da função $f(t) = e^{-\pi t^2}$: $n = 8$ (acima), $n = 16$ (ao centro) e $n = 32$ (abaixo). Os símbolos $*$'s correspondem à DTF e o traço contínuo em vermelho representa a transformada de Fourier contínua $F(\omega)$.

Algumas vezes possuimos uma amostra não igualmente espaçada e para podermos calcular a DFT precisamos fazer uma reamostragem uniforme. No próximo capítulo discutiremos métodos para obtermos tal reamostragem.

Capítulo 2

Reamostragem Uniforme

É comum ser preciso reconstruir sinais cujas amostras não estão uniformemente espaçadas. Isso é devido ao fato que a maioria dos algoritmos, baseados na Transformada de Fourier Discreta (DFT), necessitam que as amostras estejam em uma grade Cartesiana regular (uniforme). Dizemos então que é necessário que se faça uma reamostragem uniforme, i.e., interpolar as amostras não uniformes de um certo sinal em um conjunto de pontos igualmente espaçados. Utilizando o Teorema de Shannon (Cap. 1), o problema de reamostragem uniforme pode ser formulado como um sistema de equações lineares.

2.1 Caso Unidimensional

Consideremos uma função real contínua f, amostrada em um conjunto finito de pontos não igualmente espaçados, $\{\tau_1, \tau_2, \ldots, \tau_m\}$, $\tau_i \in \mathbb{R}$, $i \in M = \{1, 2, \ldots, m\}$. O problema de reamostragem uniforme consiste em encontrar uma aproximação para a função em um conjunto de pontos uniformemente espaçados, i.e., aproximar $f(t_j)$, $t_j = t_0 + j\Delta t$, $t_0 \in \mathbb{R}$, $j \in N = \{1, 2, \ldots, n\}$. A Figura 2.1 resume este problema: os símbolos $\circ$'s correspondem às amostras, e os $\times$'s representam a reamostragem uniforme.

Tal problema pode ser resolvido utilizando o Teorema de Shannon, uma vez que

$$f(\tau_i) \approx \sum_{j=1}^{n} f(t_j)\ \text{sinc}\left(\frac{\tau_i - t_j}{\Delta t}\right),\ i \in M, \tag{2.1}$$

desde que sejam satisfeitas as hipóteses do teorema.

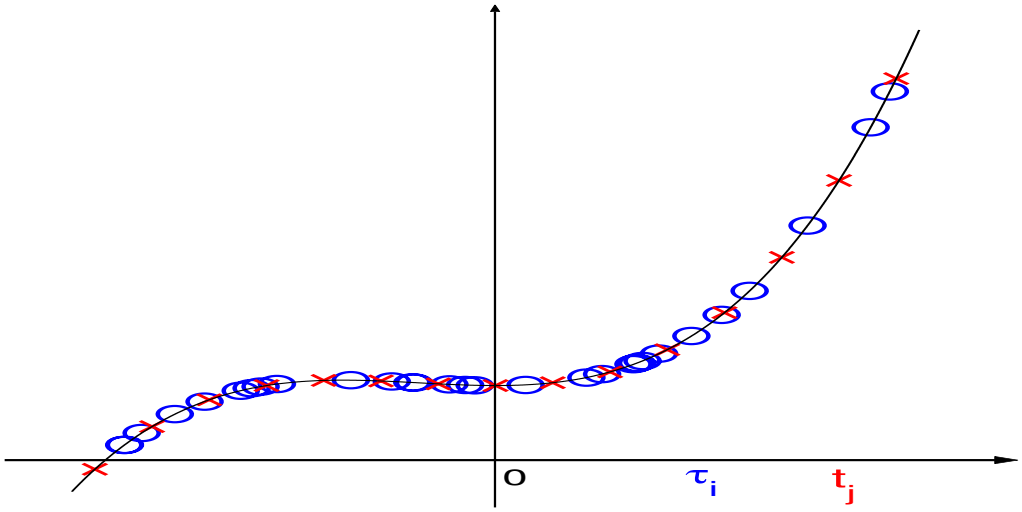


Figura 2.1: Reamostragem uniforme do sinal $s(t) = x^3 + x^2 + 2$ no intervalo $[-2, 2]$ em 16 pontos a partir de uma amostra de 32 pontos não igualmente espaçados. Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $s(t)$.

Ao tomarmos a equação (2.1) para cada $i \in M$, obtemos um sistema de equações lineares

$$Ax = b, \tag{2.2}$$

onde os elementos da matriz $A \in \mathbb{R}^{m\times n}$, do vetor $x \in \mathbb{R}^n$ e do vetor $b \in \mathbb{R}^m$ são dados, respectivamente, por $a_{ij} = \text{sinc}[(\tau_i - t_j)/\Delta t]$, $x_j = f(t_j)$ e $b_i = f(\tau_i)$, $i \in M$, $j \in N$. A matriz A é denominada matriz dos coeficientes da interpolação sinc.

2.2 A Reamostragem Uniforme e a Transformada de Fourier

Com o objetivo de constatar a utilidade da reamostragem uniforme, tomamos amostras não uniformes de 128 pontos aleatórios no intervalo $[-\pi, \pi]$, provindos de uma distribuição uniforme, dos sinais $s(t) = \text{sen}t$, $s(t) = e^{-\pi t^2}$ e $s(t) = 1$, e calculamos uma aproximação para as suas transformadas de Fourier (DFT), conforme discutido no Capítulo 1. Em seguida, reamostramos os sinais em 64 pontos igualmente espaçados, neste mesmo intervalo, e calculamos novamente a DFT em cada caso.

Nas Figuras 2.2, 2.4 e 2.6 ilustramos as amostras originais e sua reamostragens. Os resultados obtidos para as aproximações das transformadas de Fourier de tais sinais podem ser vistos nas Figuras 2.3, 2.5 e 2.7. Os resultados obtidos quando utilizou-se a reamostragem são muito

superiores aos obtidos utilizando as amostras originais. Tendo comprovado a necessidade de se fazer reamostragem uniforme, apresentamos em seguida métodos que fornecem soluções para tal problema.

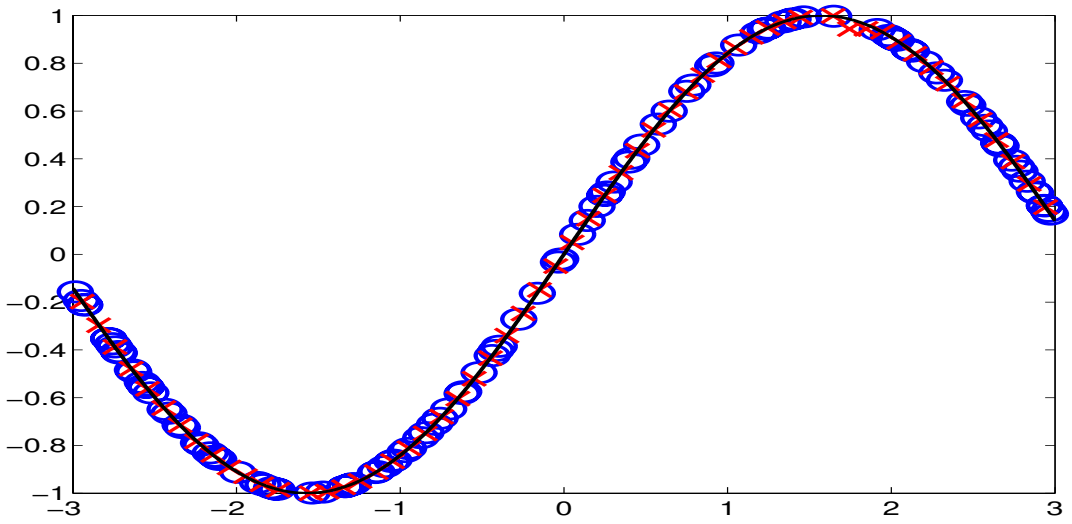


Figura 2.2: Reamostragem uniforme do sinal $s(t) = \operatorname{sen} t$. Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $s(t)$.

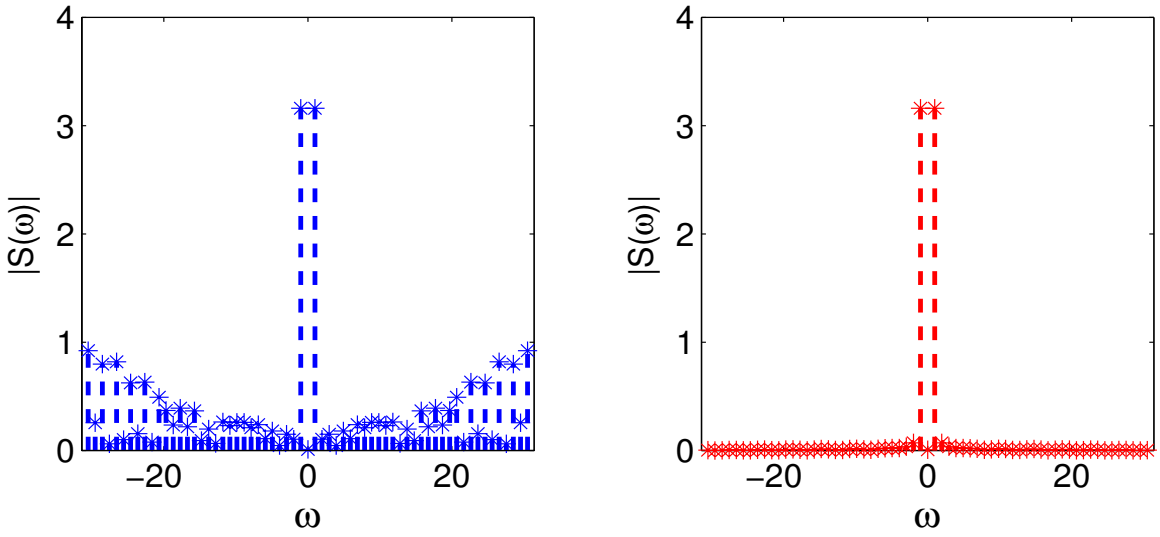


Figura 2.3: DFT do sinal $s(t) = \operatorname{sen}(t)$, em valores absolutos, utilizando uma amostra de pontos não igualmente espaçados (à esquerda) e utilizando a reamostragem uniforme da amostra original (à direita).

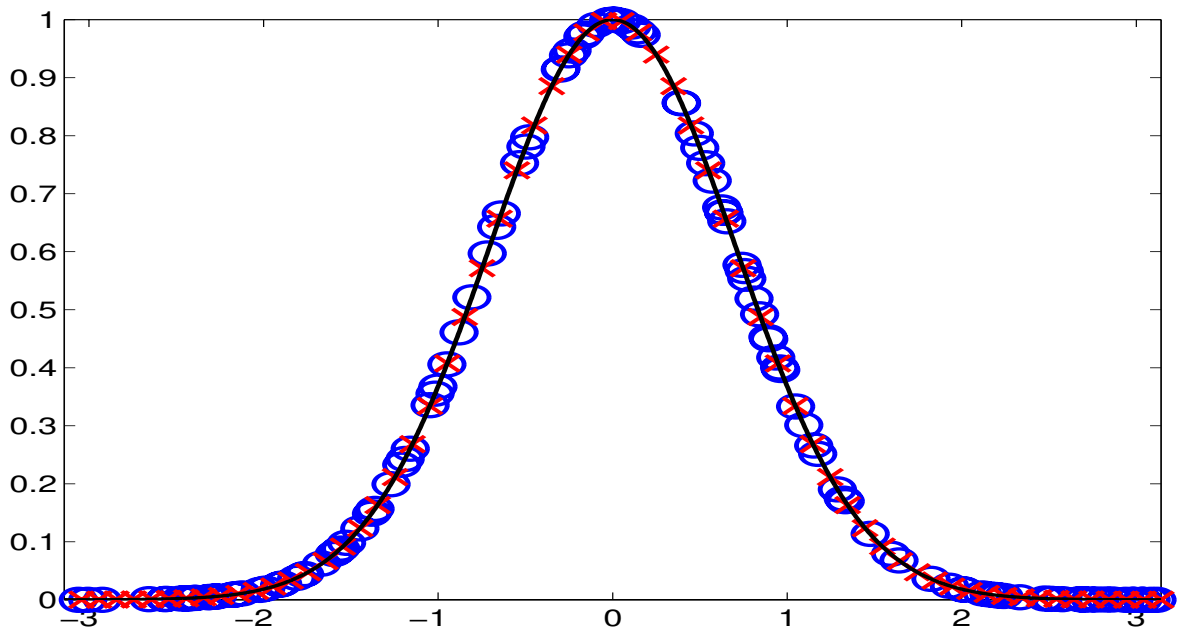


Figura 2.4: Reamostragem uniforme do sinal $s(t) = e^{-\pi t^2}$. Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $s(t)$.

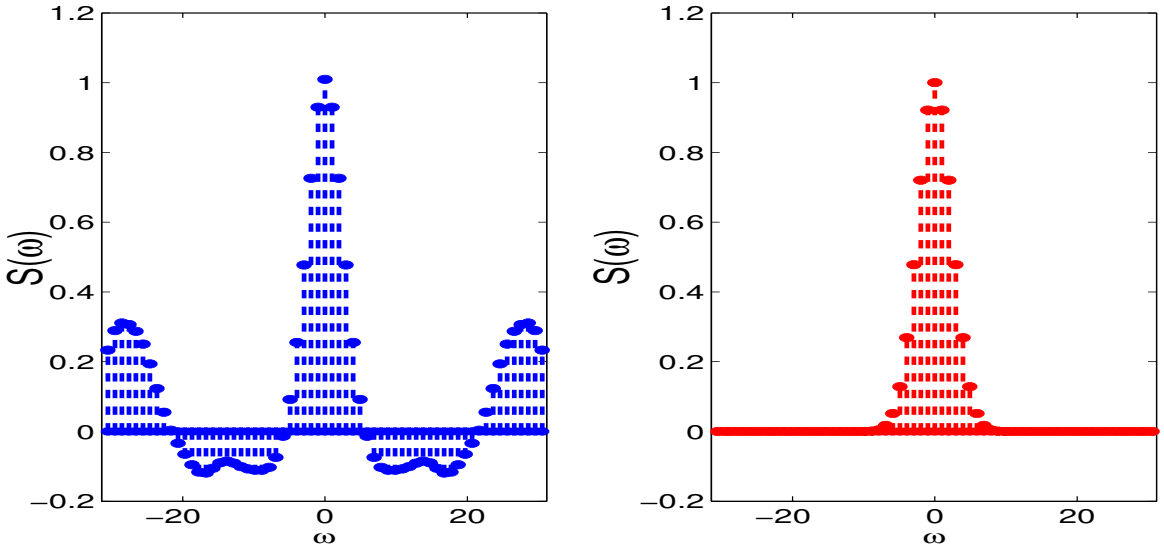


Figura 2.5: DFT do sinal $s(t) = e^{-\pi t^2}$, utilizando uma amostra de pontos não igualmente espaçados (à esquerda) e utilizando a reamostragem uniforme da amostra original (à direita).

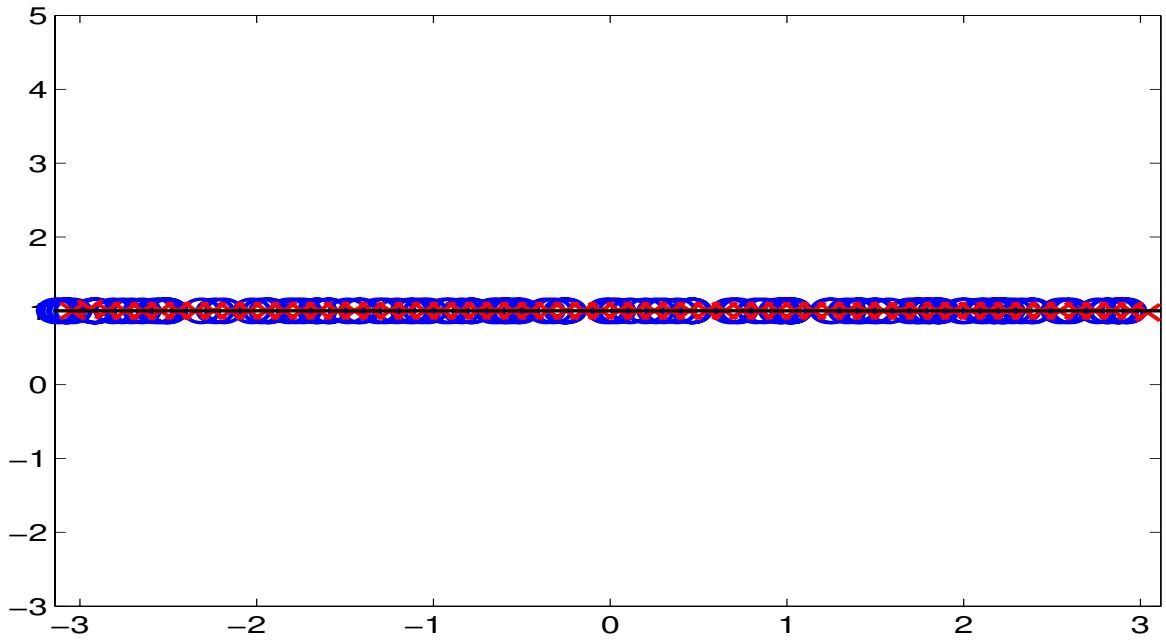


Figura 2.6: Reamostragem uniforme do sinal $s(t) = 1$. Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $s(t)$.

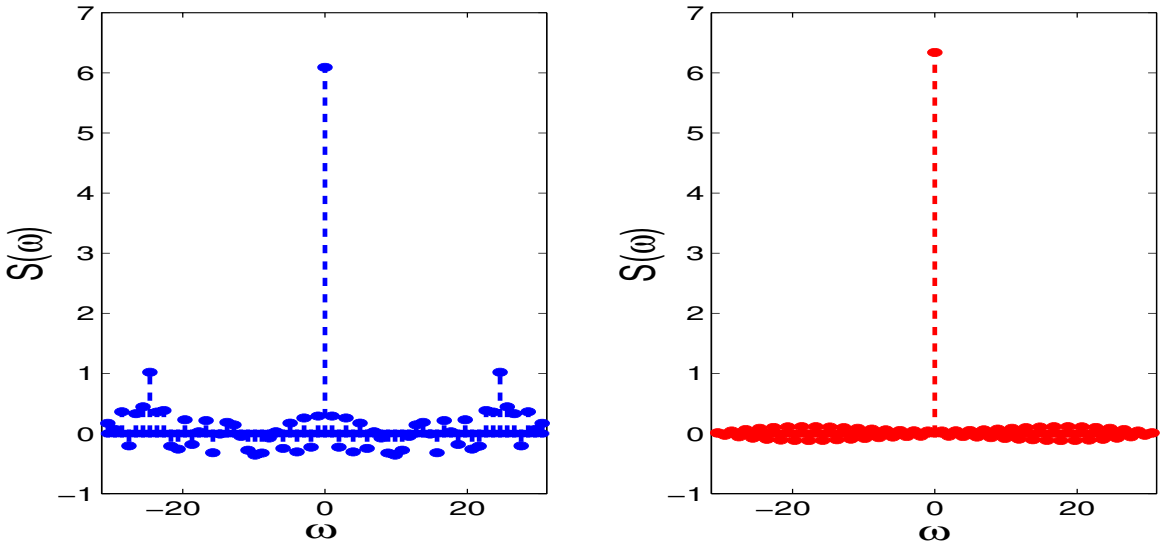


Figura 2.7: DFT do sinal $s(t) = 1$, utilizando uma amostra de pontos não igualmente espaçados (à esquerda) e utilizando a reamostragem uniforme da amostra original (à direita).

2.3 Reamostragem Uniforme Simples

Como em geral a matriz A do sistema (2.2) não é quadrada, sendo na maioria das vezes $m > n$, uma solução para este sistema pode ser encontrada utilizando a pseudoinversa (Moore-Penrose) da matriz A,

$$x = A^{+}b. \tag{2.3}$$

A matriz A^{+} tem n linhas e m colunas e satisfaz as relações $A^{+}AA^{+} = A^{+}$ e $AA^{+}A = A$. A pseudoinversa fornece uma solução ótima para a equação (2.2) no sentido de quadrados mínimos de norma mínima, i.e., ela seleciona entre todos os vetores x que minimizam a expressão $\|Ax - b\|^2$ aquele de norma mínima. Aqui, $\|\cdot\|$ denota a norma euclidiana, $\|x\| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$. Se A tem posto completo, a solução para o problema de quadrados mínimos é única e a matriz A^{+} é dada por $(A^T A)^{-1}A^T$, sendo $m > n$ (Trefethen and Bau, 1997).

O algoritmo *Reamostragem Uniforme Simples*, o qual denomina-se *URS* (do Inglês, *"Uniform ReSampling"*), computa a solução $x_{URS} = A^{+}b$ para o problema de reamostragem uniforme utilizando a Decomposição em Valores Singulares (SVD, do Inglês, *"Singular Value Decomposition"*). Apesar de x_{URS} ser, de certa maneira, uma solução ótima, temos dois problemas inerentes. Primeiro, o cálculo de A^{+} torna-se impraticável quando as dimensões são grandes demais. Quando m e n são da ordem de algumas centenas, a inversão é praticável. Segundo, cada amostra uniforme, denominada x_j, é calculada multiplicando a $j-$ésima linha de A^{+} pelo vetor b, i.e., m multiplicações e $m - 1$ adições são envolvidas. Usando o fato que a contribuição de valores que são distantes do ponto t_j é pequena, i.e., possuem coeficientes com magnitude pequena, Rosenfeld (1998) desenvolveu um algoritmo que inclui apenas um número limitado de termos em seu cálculo. Na seção seguinte este algoritmo é desenvolvido.

2.4 Reamostragem Uniforme por Blocos

Rosenfeld (1998) desenvolveu um algoritmo para encontrar uma solução da forma $x = \bar{A}^{+}b$, onde a matriz $\bar{A}^{+}$ aproxima a matriz A^{+} e cada uma de suas linhas possui na maior parte elementos nulos e somente um número restrito de elementos diferentes de zero, concentrados na vizinhança da coluna correspondente a t_j. A motivação teórica que justifica este método é discutida mais adiante, juntamente com a discussão sobre o cálculo da pseudoinversa de A. Os passos do algoritmo *Reamostragem Uniforme por Blocos*, denominado *(BURS)* (do Inglês, *"Block Uniform ReSampling"*) são como seguem: para cada t_j, ao invés de considerarmos todos os m pontos não uniformes, selecionamos apenas $\bar{m}$ pontos na vizinhança de t_j; por exemplo,

poderíamos incluir todos os pontos à uma distância δ de t_j. Similarmente, selecionamos apenas $\bar{n}$ amostras uniformes. Por exemplo, selecionamos todos os pontos igualmente espaçados, à uma distância Δ de t_j. Tudo isso é feito baseado na idéia de considerar apenas pontos que forneçam coeficientes não tão pequenos na interpolação. Rosenfeld (1998), depois de fazer alguns testes, sugeriu a relação $\Delta \geq 1.5\delta$. Utilizando apenas os dados selecionados, obtemos o seguinte sistema de equações

$$B_j x^j = b^j, \tag{2.4}$$

onde B_j é uma matriz de coeficientes de interpolação $\bar{m} \times \bar{n}$ (uma submatriz de A), $x^j \in I\!R^{\bar{n}}$ é um subvetor de x e $b^j \in I\!R^{\bar{m}}$ é um subvetor de b, os quais contém as medidas participantes. A solução da equação (2.4) não é calculada, apenas a pseudoinversa da matriz B_j, mais uma vez utilizando a SVD.

O próximo passo é isolar a linha de B_j^+ que corresponde a t_j. Esta linha contém $\bar{m}$ elementos, que são os coeficientes da combinação linear que fazem de x_j ótimo (no sentido de quadrados mínimos de norma mínima). Estes valores são agora inseridos em posições apropriadas na matriz $\bar{A}^+$. Isto é, a $j-$ésima linha da matriz $\bar{A}^+$ possui apenas elementos nulos, com excessão destes $\bar{m}$ coeficientes, os quais encontram-se em posições correspondentes às suas medidas respectivas. Isto é feito para cada coordenada uniforme t_j: o resultado é uma matriz $\bar{A}^+$, de dimensões $n \times m$, que possui a maioria dos elementos nulos, exceto por uma banda estreita ao longo de sua "diagonal". O esquema seguinte resume este método:

ALGORITMO BURS

- Dados $f(\tau_i),\ \tau_i \in I\!R,\ i \in M = \{1, 2, \ldots, m\}, t_j = t_0 + j\Delta t,\ t_0 \in I\!R,$ $j \in N = \{1, 2, \ldots, n\}, \delta > 0$ e $\Delta \geq 1.5\delta$.
- Para $j = 1, \ldots, n$ faça
  - Sejam $d = \{k \in \mathrm{M} \mid |\tau_k - t_j| < \delta\}$ e $D = \{l \in \mathrm{N} \mid |t_l - t_j| < \Delta\}$.
  - Seja B_j a submatriz de A composta pelos elementos $a_{kl},\ k \in d,\ l \in D$.
  - Compute B_j^+.
  - Isole a linha de B_j^+, que corresponde a t_j, e insira-a em locais apropriados na $j-$ésima linha de $\bar{A}^+$.
- Seja $b = (f(\tau_1), \ldots, f(\tau_m))^T$.
- As amostras uniformes são dadas por $x_{BURS} = \bar{A}^+ b$, i.e., $f(t_j) \approx (x_{BURS})_j$.

Tal algoritmo é muito eficiente porque ele troca o problema de calcular a pseudoinversa de uma matriz grande por calcular n pseudoinversas menores. Além disso, a maioria dos elementos da matriz $\bar{A}^+$ é zero e assim, a solução dada por x_{BURS}, é menos custosa computacionalmente.

2.5 Decomposição em Valores Singulares

A *Decomposição em Valores Singulares* (SVD) é uma fatoração de matrizes utilizada em muitos algoritmos e também para fins conceituais. É uma ferramenta contida na maioria dos pacotes matemáticos de computação e muitos problemas de Álgebra Linear podem ser resolvidos utilizando esta fatoração, dentre eles o cálculo da pseudoinversa de uma matriz (e.g. Klema and Laub (1980)).

DEFINIÇÃO 2.1 *Seja $A \in I\!R^{m\times n}$. A SVD de A é a fatoração*

$$A = U\Sigma V^T \tag{2.5}$$

onde $U \in I\!R^{m\times m}$ e $V \in I\!R^{n\times n}$ são ortogonais e $\Sigma \in I\!R^{m\times n}$ é tal que $\sigma_{ij} = 0$, para $i \neq j$. Os elementos σ_{ii}, para $i = 1, 2, \ldots, p = \min\{m, n\}$ são denominados valores singulares de A e denotados por σ_i, sendo escolhidos de modo que $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$.

As m colunas da matriz U, u_k, são chamadas vetores singulares à esquerda de A e as n colunas da matriz V, v_k, são chamadas vetores singulares à direita de A. Os vetores u_k são os autovetores da matriz A^TA, e os vetores v_k são os autovetores da matriz AA^T. Também temos que σ_k^2 são os autovalores das matrizes A^TA e AA^T. Note que sempre a matriz Σ tem as mesmas dimensões da matriz A, enquanto que as matrizes U e V são quadradas e ortogonais.

DEFINIÇÃO 2.2 *Seja $A \in I\!R^{m\times n}$, $m \geq n$, e $A = U\Sigma V^T$ a SVD de A. Devido aos zeros da matriz Σ, apenas as n primeiras colunas da matriz U influenciam na fatoração da matriz A. Assim,*

$$A = \widehat{U}\widehat{\Sigma}V^T, \tag{2.6}$$

onde $\widehat{U} \in I\!R^{m\times n}$ é formada pelas primeiras colunas da matriz U, acima mencionadas, e $\widehat{\Sigma} \in I\!R^{n\times n}$ é diagonal, formada pelas n primeiras linhas da matriz Σ. Essa fatoração da matriz A é conhecida como a SVD reduzida de A.

TEOREMA 2.1 *Toda matriz $A \in I\!R^{m\times n}$ possui uma decomposição em valores singulares.*

PROVA: Ver Trefethen and Bau (1997) e Golub and Loan (1996). □

2.6 A SVD e o Cálculo da Pseudoinversa

Nas seções anteriores discutimos dois métodos que utilizam a SVD para encontrar uma solução para problema de reamostragem uniforme, ou seja, para o sistema (2.2). Veremos agora com detalhes como esta ferramenta pode ser útil para o nosso problema. Vamos supor que a matriz $A \in I\!R^{m\times n},\ m > n$, tenha posto completo. Então, como vimos anteriormente, a pseudoinversa da matriz A é única e dada por

$$A^+ = (A^T A)^{-1} A^T. \tag{2.7}$$

É conveniente decompor a matriz A^+ utilizando a SVD reduzida da matriz A, $A = \widehat{U}\widehat{\Sigma}V^T$. Substituindo na equação acima, obtemos

$$A^+ = V\widehat{\Sigma}^{-1}\widehat{U}^T, \tag{2.8}$$

e, portanto, a solução de quadrados mínimos pode ser escrita como

$$x = A^+ b = V\widehat{\Sigma}^{-1}\widehat{U}^T b. \tag{2.9}$$

Considerando o "sistema singular" $\{u_k,\ v_k;\ \sigma_k\}$, formado pela k-ésima coluna de $\widehat{U}$, a k-ésima coluna de V e o k-ésimo valor singular de A, a solução de quadrados mínimos pode ser escrita como

$$x = A^+ b = \sum_{k=1}^{n} \sigma_k^{-1}(v_k^T b)u_k. \tag{2.10}$$

Analisando a equação (2.10), podemos notar que, à medida que k aumenta, σ_k^{-1} aumenta também, e, pequenas perturbações que possam ocorrer são multiplicadas por fatores cada vez maiores. Isto significa que perturbar o vetor b de um pequeno fator, implicará em perturbações muito grandes na solução x, o que implica que este problema seja classificado como mal condicionado. Isto é piorado quanto maior o valor de n, o que justifica o fato do algoritmo URS não ser praticável quando os valores de m e n são muito grandes.

2.7 Regularização

A regularização consiste em tornar a solução para um determinado problema mais estável, modificando-o de maneira que a nova solução seja menos sensível à pequenas perturbações nos dados. Ao mesmo tempo, a solução do problema modificado deve ser próxima à solução

do problema original. O método consiste em definir um operador R_ρ, parametrizado por uma parâmetro positivo de suavidade ρ, que aproxima a solução dada pela pseudoinversa da seguinte maneira

$$\lim_{\rho \to 0^+} R_\rho b = A^+ b. \tag{2.11}$$

Assim, a solução original $x = A^+ b$ é substituida pela solução aproximada $\tilde{x} = R_\rho b$. O parâmetro ρ afeta tanto a precisão do método quanto a estabilidade, quanto menor for ρ, mais próxima da solução original fica a solução aproximada, e, quanto maior o valor deste parâmetro, mais estável a solução. Desta maneira, este parâmetro deve ser escolhido de maneira a balancear estabilidade e precisão.

Apresentamos agora um tipo de técnica de regularização chamada *janelamento espectral*. Neste método, a equação (2.10) é substituída por uma versão filtrada

$$\tilde{x} = R_\rho b = \sum_{k=1}^{n} r_k(\rho) \sigma_k^{-1} (v_k^T b) u_k. \tag{2.12}$$

onde as funções $r_k : [0, \infty) \to [0, 1]$ satisfazem: $\lim_{\rho \to 0^+} r_k(\rho) = 1$ para todo $k = 1, 2, \ldots, n$ e $\lim_{k \to \infty} r_k(\rho)\ \sigma_k^{-1} = 0$. A última condição assegura que os termos instáveis, de alta freqüência, sejam eliminados, enquanto que a primeira assegura que a equação (2.11) seja satisfeita.

Existem muitos regularizadores tipo "janelamento espectral" para o problema de quadrados mínimos usando SVD, aqui descreveremos dois deles. O primeiro é conhecido como *expansão para o sistema singular truncada*. Neste método, os "pesos" $r_k(\rho)$ são dados por

$$r_k(\rho) = \begin{cases} 1, & k < [1/\rho] \\ 0, & \text{caso contrário.} \end{cases} \tag{2.13}$$

onde $[1/\rho]$ designa o maior inteiro menor que $1/\rho$. Como no algoritmo BURS utilizamos apenas uma vizinhança em torno de cada ponto onde se quer conhecer a função, e então calculamos uma pseudoinversa aproximada, podemos entendê-lo como se truncasse a SVD da matriz, por isso ele é um exemplo para este tipo de "janelamento espectral".

Este algoritmo é muito preciso, porém, muito sensível à ruído. Mesmo a menor presença deste, corrompe a imagem gerada pelo BURS. Por isso Rosenfeld (2002) desenvolveu um outro algoritmo, o qual denominou *rBURS*. Antes de descrevermos tal algoritmo, apresentamos outro tipo de regularização "janelamento espectral" chamado *Filtro Tikhonov*, ao qual este novo algoritmo faz parte (Tikhonov and Arsenin, 1977).

Este método tem como operador de regularização, a matriz

$$R_\rho = (A^T A + \rho I)^{-1} A^T, \quad \rho > 0, \tag{2.14}$$

ou seja, fornece a solução regularizada

$$\tilde{x} = R_\rho b = (A^T A + \rho I)^{-1} A^T b, \tag{2.15}$$

que equivale a minimizar a soma $\|Ax - b\|^2 + \rho\|x\|$ e cuja solução é conhecida como uma versão regularizada da equação normal $A^T Ax = A^T b$ (ver Lawson and Hanson (1974) e Björck (1996)), a qual dá origem ao problema de quadrados mínimos e conseqüentemente à matriz A^+. Se tomarmos a fatoração SVD da matriz A e substituirmos na equação (2.14), obtemos

$$R_\rho = V[(\Sigma^T \Sigma + \rho I)^{-1} \Sigma^T] U^T. \tag{2.16}$$

Como todas as matrizes dentro do colchetes são matrizes diagonais, com valores reais ao longo da diagonal já que a matriz A é real, não é necessário fazer inversão de nenhuma matriz, obtendo a seguinte estrutura

$$R_\rho = V \begin{bmatrix} \Delta & 0 \\ 0 & 0 \end{bmatrix} U^T, \tag{2.17}$$

onde Δ é uma matriz diagonal $p \times p$ com os seguintes elementos em sua diagonal $\{\sigma_1/(\sigma_1^2 + \rho)$, $\sigma_2/(\sigma_2^2 + \rho), \ldots, \sigma_p/(\sigma_p^2 + \rho), \}$, sendo $\sigma_1, \sigma_2, \ldots, \sigma_p$ os valores singulares de A. Assim, neste método os pesos são dados por

$$r_k(\rho) = \frac{\sigma_k^2}{\sigma_k^2 + \rho} \tag{2.18}$$

e tornam a solução menos sensível à presença de ruídos. Note que os pesos satisfazem as três condições descritas anteriormente.

2.8 Reamostragem Uniforme por Blocos Regularizada

Consideremos uma amostra de uma função contínua f nos pontos $\{\tau_1, \tau_2, \ldots, \tau_m\}$, $\tau_i \in I\!R$, $i \in M = \{1, 2, \ldots, m\}$. Nosso objetivo é encontrar uma aproximação para a função nos pontos uniformemente espaçados, $t_j = t_0 + j\Delta t$, $t_0 \in I\!R$, $j \in N = \{1, 2, \ldots, n\}$. Seja $Ax = b$ a formulação para tal problema, conforme foi visto nas seções anteriores.

Foram apresentados dois algoritmos, URS e BURS, que fornecem uma solução para tal

problema, porém, devido à ineficiência do algoritmo BURS em problemas em que é encontrado algum tipo de ruído, Rosenfeld (2002) desenvolveu uma versão regularizada deste algoritmo e a denominou de rBURS. Neste algoritmo, também é selecionada uma submatriz B_j da matriz A para cada amostra uniforme t_j, e calculada a sua pseudoinversa com o objetivo de selecionar a linha correspondente a t_j e inseri-la na j-ésima linha da matriz $\bar{A}^+$, que aproxima a pseudoinvesa A^+. O que altera é o cálculo da pseudoinversa B_j^+. Ao invés de a calcularmos, também a aproximamos pela versão regularizada $\bar{B}_j^+ = (B_j^T B_j + \rho I)^{-1} B_j^T$ que é calculada usando a SVD, como descrito na seção anterior. Note que o BURS corresponde ao rBURS com $\rho = 0$. A seguir temos um esquema que resume este método:

ALGORITMO RBURS

- Dados $f(\tau_i),\ \tau_i \in I\!R,\ i \in M = \{1, 2, \ldots, m\}, t_j = t_0 + j\Delta t,\ t_0 \in I\!R,$ $j \in N = \{1, 2, \ldots, n\}, \delta > 0$ e $\Delta \geq 1.5\delta$.
- Para $j = 1, \ldots, n$ faça
  - Sejam $d = \{k \in \mathrm{M} \mid |\tau_k - t_j| < \delta\}$ e $D = \{l \in \mathrm{N} \mid |t_l - t_j| < \Delta\}$.
  - Seja B_j a submatriz de A composta pelos elementos $a_{kl},\ k \in d,\ l \in D$.
  - Compute $\bar{B}_j^+ = (\bar{B}_j^T \bar{B}_j + \rho I)^{-1} \bar{B}_j^T$ usando SVD.
  - Isole a linha de $\bar{B}_j^+$, que corresponde a t_j, e insira-a em locais apropriados na $j-$ésima linha de $\bar{A}^+$.
- Seja $b = (f(\tau_1), \ldots, f(\tau_m))^T$.
- As amostras uniformes são dadas por $x_{BURS} = \bar{A}^+ b$, i.e., $f(t_j) \approx (x_{BURS})_j$.

Note que, no i-ésimo passo do rBURS, queremos apenas a linha da matriz $\bar{B}_j^+$ que corresponde à t_j, como no BURS. Supondo que esta linha seja a q-ésima linha de tal matriz, não precisamos calcular toda a matriz $\bar{B}_j^+$, já que dada a fatoração SVD da matriz $B_j = U\Sigma V^T$, temos

$$(\bar{B}_j^+)_q = V_q \begin{bmatrix} \Delta & 0 \\ 0 & 0 \end{bmatrix} U^T, \tag{2.19}$$

onde V_q é a q-ésima linha da matriz V, e a matriz Δ é como descrita na seção anterior.

Exemplos da eficiência deste algoritmo e comparações com os outros dois algoritmos aqui desenvolvidos são apresentadas no Capítulo 3.

2.9 Método dos Gradientes Conjugados

O método dos Gradientes Conjugados (CG, do Inglês, *"Conjugate Gradient"*) é um método iterativo e uma ferramenta padrão para resolução de problemas de quadrados mínimos. Podemos então, utilizá-lo na resolução do problema de reamostragem uniforme, uma vez que este problema pode ser formulado com um sistema $Ax = b$.

O método dos gradientes conjugados é um método para sistemas lineares simétricos e positivos definidos. A princípio, todo método iterativo deste tipo pode ser aplicado ao sistema de equações normais $A^T Ax = A^T b$.

Neste método, calculamos apenas os produtos das matrizes A e A^T por um vetor, não sendo necessário calcular $A^T A$. Trabalhar somente com A e A^T tem uma vantagem importante. O cálculo do produto $A^T A$, como foi visto, torna a solução sensível à pequenas perturbações nos dados. Aqui, damos uma aproximação inicial para a solução e ela é melhorada até que seja obtida uma solução aceitável. Para isso, a matriz A não precisa necessariamente ser conhecida, apenas a ação de A e A^T em vetores. Assim, este método, a cada iteração, encontra uma aproximação para a solução do sistema, o resíduo correspondente a esta aproximação e procura uma direção usada na atualização destes. Vejamos agora como isto é feito.

DEFINIÇÃO 2.3 *Dada uma matriz $B \in \mathbb{R}^{n\times n}$ e um vetor $c \in \mathbb{R}^n$, o subespaço de Krylov $\mathcal{K}_k(B, c)$ é dado por*

$$\mathcal{K}_k(B, c) = span\{c, Bc, \ldots, B^{k-1}c\}, \text{ para algum } k \geq 1. \tag{2.20}$$

Na resolução do sistema $Ax = b$, a k-ésima iteração do método CG encontra uma aproximação x^k para a solução $x = A^+ b$. Esta iteração é unicamente determinada pela seguinte propriedade: x^k minimiza

$$E(x) = \|Ax - b\|^2 \tag{2.21}$$

no subespaço afim $x^0 + \mathcal{K}_k(A^T A, s^0)$, onde

$$s^0 = -\nabla E(x^0) = A^T(b - Ax^0). \tag{2.22}$$

Maiores detalhes podem ser vistos em Björck (1996) e Golub and O'Leary (1989).

ALGORITMO CG

- Dada uma aproximação inicial x^0, sejam $r^0 = b - Ax^0$,
  $p^0 = s^0 = A^T r^0$, $\psi_0 = \|s^0\|^2$.
- Para $k = 0, 1, \ldots$, enquanto $\psi_k > tol$, faça
  $q^k = Ap^k$,
  $\alpha_k = \psi_k / \|q^k\|^2$,
  $x^{k+1} = x^k + \alpha_k p^k$,
  $r^{k+1} = r^k - \alpha_k q^k$,
  $s^{k+1} = A^T r^{k+1}$,
  $\psi_{k+1} = \|s^{k+1}\|^2$,
  $\beta_k = \psi_{k+1} / \psi_k$,
  $p^{k+1} = s^{k+1} + \beta_k p^k$.

Como é considerado um método rápido e eficiente para resolução do problema de quadrados mínimos, no próximo capítulo também formulamos exemplos de reamostragem uniforme utilizando CG, com o objetivo de comparar com os resultados dos algoritmos URS e BURS.

2.10 Caso Bidimensional

Consideremos uma função real contínua $f : \mathbb{R}^2 \to \mathbb{R}$, amostrada em um conjunto finito de pontos não igualmente espaçados, $\{(\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots, (\alpha_m, \beta_m)\}$. Nosso objetivo é encontrar uma aproximação para $f(a_i, b_j)$, $(a_i, b_j) = (a_0, b_0) + (i\Delta a, j\Delta b)$, $(a_0, b_0) \in \mathbb{R}^2$, $i \in N_1 = \{1, 2, \ldots, n_1\}$ e $j \in N_2 = \{1, 2, \ldots, n_2\}$.

Tal problema pode ser resolvido aplicando-se o Teorema de Shannon (Cap. 1) para dimensões maiores. Para cada (α_k, β_k), $k \in M = \{1, 2, \ldots, m\}$, temos

$$f(\alpha_k, \beta_k) \approx \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} f(a_i, b_j) \text{ sinc} \left(\frac{\alpha_k - a_i}{\Delta a} \right) \text{ sinc} \left(\frac{\beta_k - b_j}{\Delta b} \right). \tag{2.23}$$

Ordenamos os pontos Cartesianos unifomemente espaçados por coluna (poderia ter sido por linha) em um vetor, como segue: $\{(\widehat{a}_1, \widehat{b}_1), (\widehat{a}_2, \widehat{b}_2), \ldots, (\widehat{a}_n, \widehat{b}_n)\}$, onde $(\widehat{a}_1, \widehat{b}_1) = (a_1, b_1)$, $(\widehat{a}_2, \widehat{b}_2) = (a_1, b_2)$, $(\widehat{a}_3, \widehat{b}_3) = (a_1, b_3), \ldots, (\widehat{a}_{n_2}, \widehat{b}_{n_2}) = (a_1, b_{n_2})$, $(\widehat{a}_{n_2+1}, \widehat{b}_{n_2+1}) = (a_2, b_1), \ldots,$

$(\widehat{a}_n, \widehat{b}_n) = (a_{n_1}, b_{n_2})$, onde $n = n_1 n_2$. Assim, a equação (2.23), pode reescrita da seguinte forma

$$f(\alpha_k, \beta_k) \approx \sum_{l=1}^{n} f(\widehat{a}_l, \widehat{b}_l) \text{ sinc} \left(\frac{\alpha_k - \widehat{a}_l}{\Delta a} \right) \text{ sinc} \left(\frac{\beta_k - \widehat{b}_l}{\Delta b} \right), \ k \in M. \tag{2.24}$$

Observemos que, como no caso unidimensional, o conjunto de equações acima formam um sistema linear $CX = D$. Os elementos da matriz $C \in I\!R^{m \times n}$, do vetor $X \in I\!R^n$ e do vetor $D \in I\!R^m$ são dados por $c_{kl} = \text{ sinc}\,[\,(\alpha_k - \widehat{a}_l)/\Delta a\,]\,\text{sinc}\,[\,(\beta_k - \widehat{b}_l)/\Delta b\,]$, $X_l = f(\widehat{a}_l, \widehat{b}_l)$, $D_k = f(\alpha_k, \beta_k)$, $k \in M$, $l \in N = \{1, 2, \ldots, n\}$. A solução para este sistema é dada por

$$X = C^{+} D, \tag{2.25}$$

e todos os métodos discutidos anteriormente no caso unidimensional podem ser aplicados. No caso dos algoritmos BURS e rBURS a utilização dos parâmetros δ e Δ no cálculo da SVD "truncada" da matriz, é baseada na norma infinito, ou seja, para cada ponto uniforme $(\widehat{a}_l, \widehat{b}_l)$, $l \in N$, selecionamos todos os pontos $(\alpha_k,\, \beta_k)$, $k \in M$, tal que $\max\{\widehat{a}_l - \alpha_k, \widehat{b}_l - \beta_k\} < \delta$, ou seja, selecionamos os pontos que estão "dentro" do quadrado de lado 2δ, centrado em $(\widehat{a}_l,\, \widehat{b}_l)$. Analogamente, selecionamos os pontos igualmente espaçados que estão "dentro" do quadrado de lado 2Δ, centrado em $(\widehat{a}_l,\, \widehat{b}_l)$. Uma vez selecionados tais pontos, os outros passos destes algoritmos são análogos ao caso unidimensional.

No próximo capítulo apresentaremos exemplos, tanto no caso unidimensional quanto no caso bidimensional, comparando a eficiência dos métodos URS e BURS em problemas diversos.

Capítulo 3

Experimentos Computacionais

Com o objetivo de validar os resultados discutidos anteriormente, formulamos alguns exemplos, no caso unidimensional e no caso bidimensional. Comparamos os resultados dos algoritmos URS, BURS e CG para o problema de reamostragem uniforme e, no caso de ineficiência do algoritmo BURS, também aplicamos o algoritmo rBURS para resolução do problema em questão.

3.1 Caso Unidimensional

Em cada um dos exemplos a seguir foram gerados m pontos aleatórios provindos de uma distribuição uniforme e n pontos igualmente espaçados, todos em um intervalo pré-determinado $[T_1, T_2]$. Tomamos então uma função amostrada nos m pontos acima e reamostramos tal função nos n pontos desejados.

3.1.1 Função Exponencial

$f(t) = e^{t/2}$, $T_1 = -3$, $T_2 = 3$, $m = 256$ e $n = 128$.

Neste primeiro exemplo, o resultado apresentado pelo algoritmo de reamostragem simples (URS) não foi satisfatório, diferentemente do apresentado pelo algoritmo de reamostragem por blocos (BURS). Fizemos alguns testes com este último, variando os parâmetros δ e Δ com o objetivo de validar a relação $\Delta \geq 1.5\delta$ proposta por Rosenfeld (1998).

O BURS foi aplicado três vezes e em cada caso tomamos $\delta = 0.3$ e variamos Δ, cujos valores utilizados foram 0.3, 0.4 e 0.5 respectivamente. Em cada caso, o algoritmo BURS substituiu o cálculo da pseudoinversa de dimensão 128×256 pelo cálculo de 128 pseudoinversas

de dimensões menores, que variaram entre 14×7 e 34×13 no primeiro caso, 14×9 e 34×17 no segundo e 14×9 e 34×21 no último caso. Os resultados analisados neste exemplo podem ser observados na Figura 3.1. Podemos notar que quando a relação acima entre os parâmetros foi satisfeita, o resultado aproximou muito bem a função, mesmo esta não sendo banda limitada. Em todos os casos analisados, o tempo de execução dos algoritmos não ultrapassou 1 segundo. O erro foi medido pela norma euclidiana da diferença entre o vetor que continha os valores exatos da função nos pontos reamostrados e o fornecido por cada algoritmo dividido pela norma do vetor com os valores exatos.

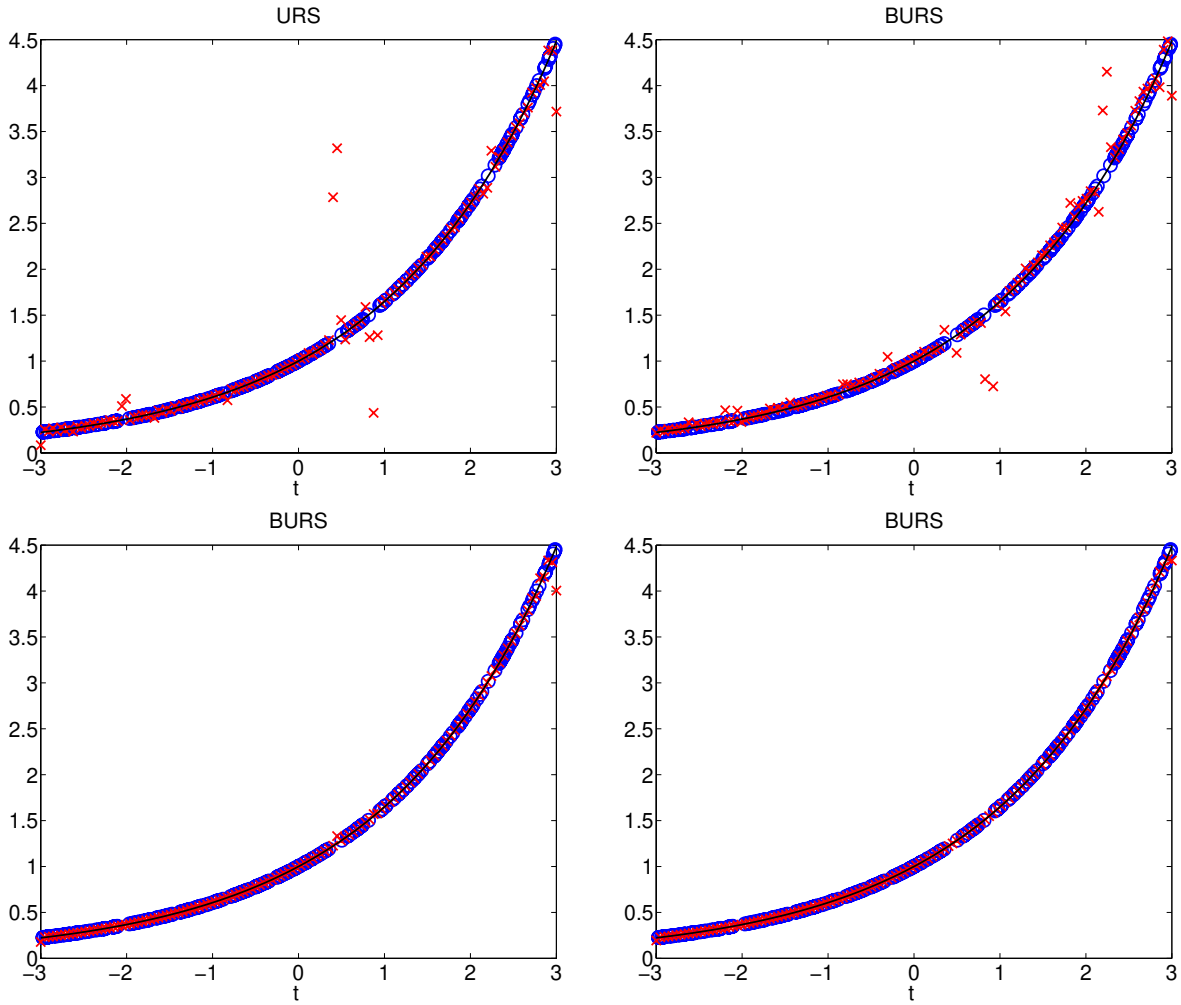


Figura 3.1: Reamostragem uniforme do sinal $f(t) = e^{t/2}$: URS, erro $= 12.83\%$ (acima à esquerda), BURS, $\delta = 0.3$, $\Delta = 0.3$, erro $= 10.55\%$ (acima à direita), BURS, $\delta = 0.3$, $\Delta = 0.4$ e erro $= 1.75\%$ (abaixo à esquerda) e BURS, $\delta = 0.3$, $\Delta = 0.5$ e erro $= 0.7\%$ (abaixo à direita). Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $f(t)$.

Em todos os próximos exemplos utilizamos como parâmetros $\delta = 5\Delta t$ e $\Delta = 1.8\delta$, onde Δt é o espaçamento das amostras uniformes. O erro, como anteriormente, foi medido pela norma euclidiana da diferença entre o vetor que continha os valores exatos da função nos pontos reamostrados e o fornecido por cada algoritmo dividido pela norma do vetor com os valores exatos.

3.1.2 Função Trigonométrica

$f(t) = 10\,\text{sen}t\,\cos 10t + \cos 3t - \text{sen}2t$, $T_1 = -3$, $T_2 = 3$.

Como esta função é uma combinação de senos e cossenos (banda limitada) os dois algoritmos devem fornecer bons resultados. Consideramos vários valores de m e n começando com $m = 64$ e $n = 32$ dobrando os valores até $m = 4096$ e $n = 2048$, com o objetivo de constatar até que ponto o algoritmo URS é praticável.

Pudemos notar que o algoritmo URS forneceu bons resultados, porém não tão bons quanto os fornecidos pelo BURS, e, em um tempo menor que o BURS até m e n da ordem de centenas, o que comprova os resultados obtidos por Rosenfeld (1998). Os resultados podem ser vistos nas Figuras 3.2 a 3.8.

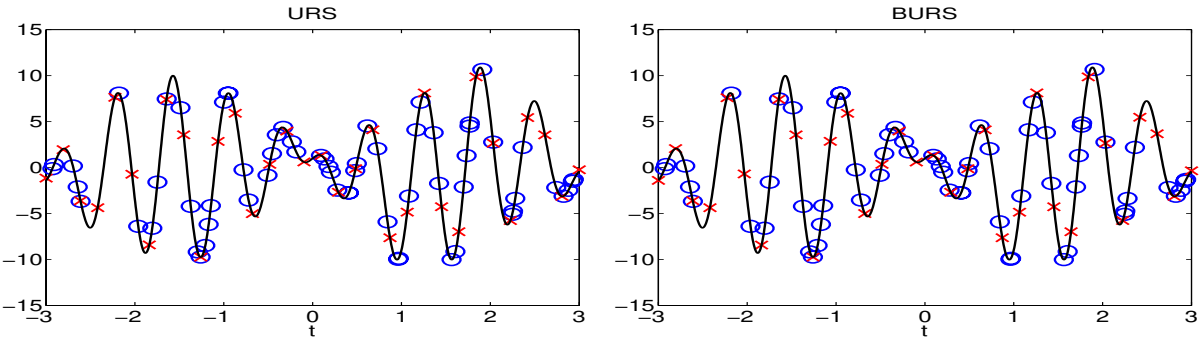


Figura 3.2: Reamostragem uniforme do sinal $f(t) = 10\,\text{sen}t\cos 10t + \cos 3t - \text{sen}2t$, $m = 64$, $n = 32$: URS, TEMPO = 0.001s, ERRO = 1.22% (à esquerda) e BURS, TEMPO = 0.032s, ERRO = 0.32% (à direita). Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $f(t)$.

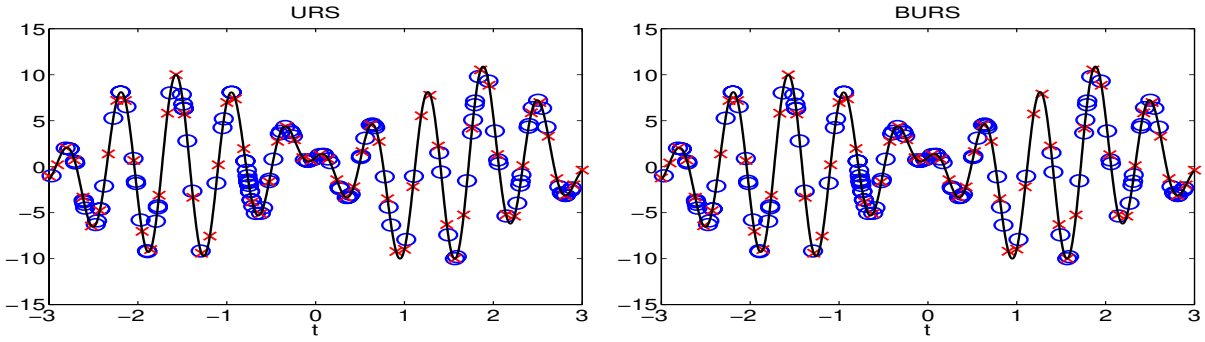


Figura 3.3: Reamostragem uniforme do sinal $f(t) = 10\,\text{sen}t\cos 10t + \cos 3t - \text{sen}2t$, $m = 128$, $n = 64$: URS, TEMPO = 0.001s, ERRO = 1.28% (à esquerda) e BURS, TEMPO = 0.125s, ERRO = 0.48% (à direita). Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $f(t)$.

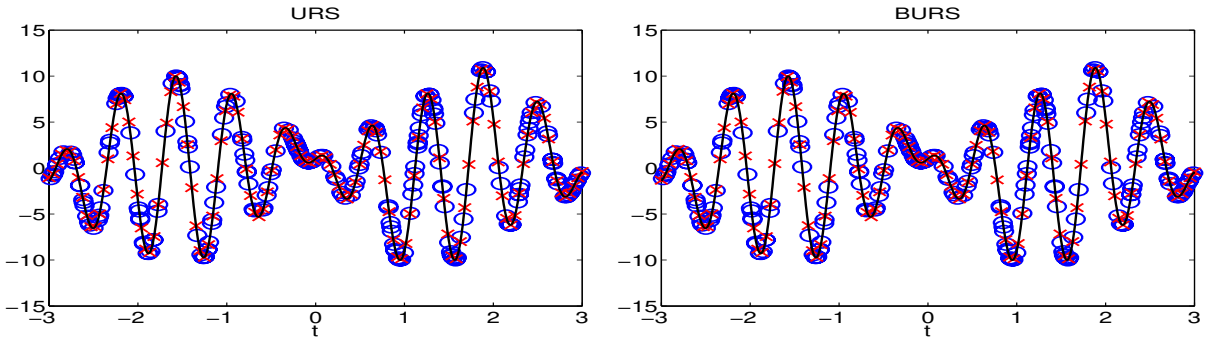


Figura 3.4: Reamostragem uniforme do sinal $f(t) = 10\,\text{sen}t\cos 10t + \cos 3t - \text{sen}2t$, $m = 256$, $n = 128$: URS, TEMPO = 0.015s, ERRO = 0.68% (à esquerda) e BURS, TEMPO = 0.328s, ERRO = 0.14% (à direita). Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $f(t)$.

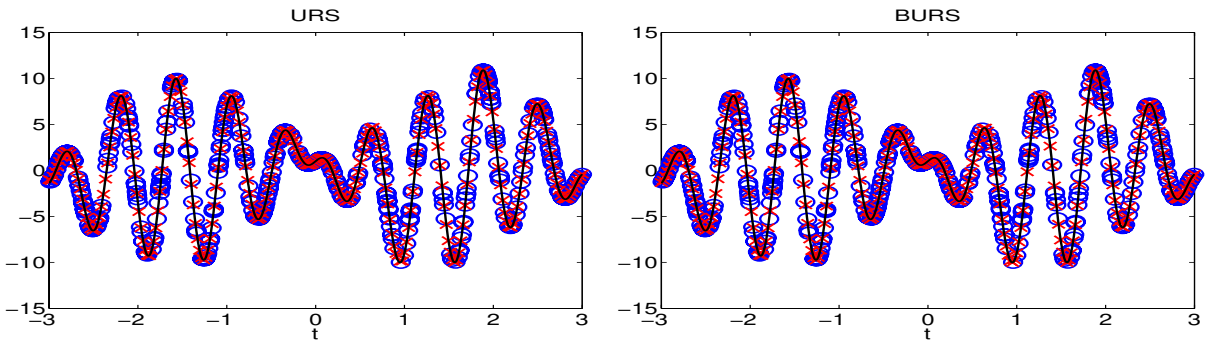


Figura 3.5: Reamostragem uniforme do sinal $f(t) = 10\,\text{sen}t\cos 10t + \cos 3t - \text{sen}2t$, $m = 512$, $n = 256$: URS, TEMPO = 0.250s, ERRO = 0.98% (à esquerda) e BURS, TEMPO = 1.031s, ERRO = 0.01% (à direita). Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $f(t)$.

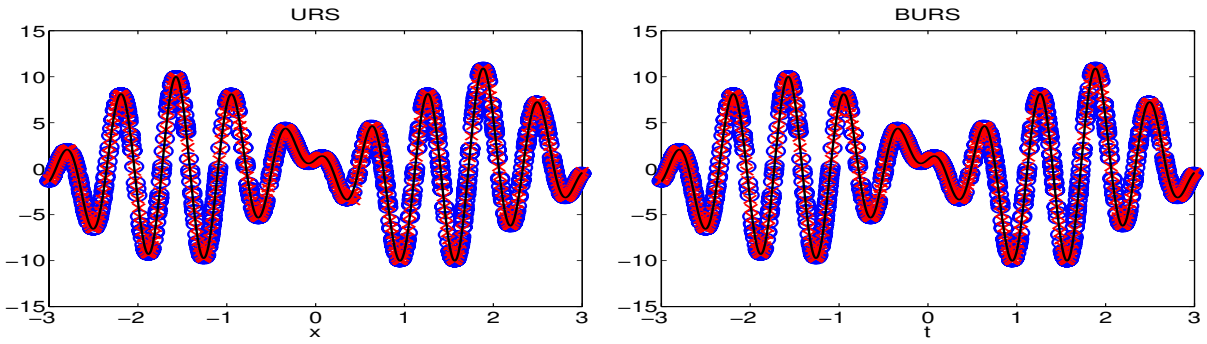


Figura 3.6: Reamostragem uniforme do sinal $f(t) = 10\,\text{sen}t\cos 10t + \cos 3t - \text{sen}2t$, $m = 1024$, $n = 512$: URS, TEMPO = 2.047s, ERRO = 1.51% (à esquerda); BURS, TEMPO = 3.875s, ERRO = 0.08% (à direita). Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $f(t)$.

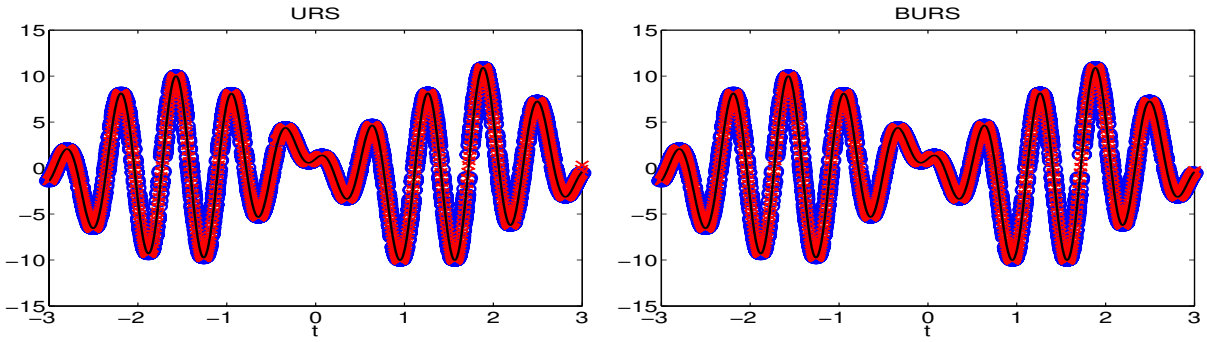


Figura 3.7: Reamostragem uniforme do sinal $f(t) = 10\,\text{sen}t\cos 10t + \cos 3t - \text{sen}2t$, $m = 2048$, $n = 1024$: URS, TEMPO = 17.063s, ERRO = 0.72% (à esquerda) e BURS, TEMPO = 14.843s, ERRO = 0.09% (à direita). Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $f(t)$.

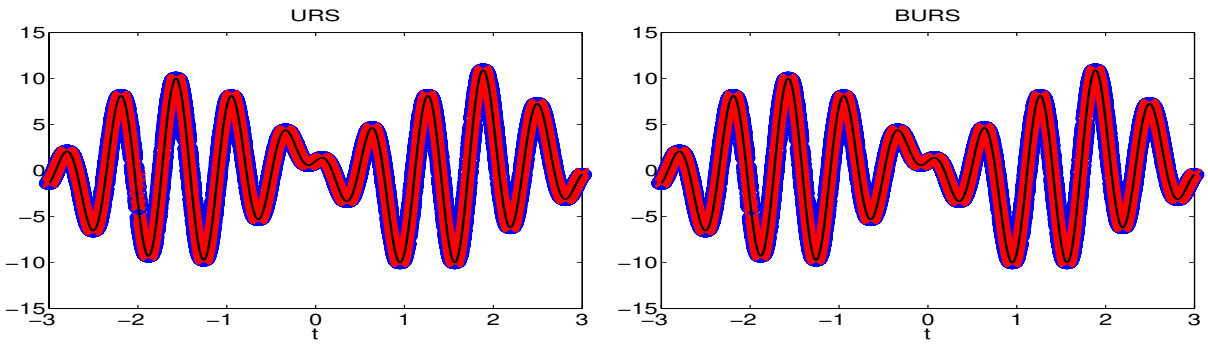


Figura 3.8: Reamostragem uniforme do sinal $f(t) = 10\,\text{sen}t\cos 10t + \cos 3t - \text{sen}2t$, $m = 4096$, $n = 2048$: URS, TEMPO = 2min26s, ERRO = 0.56% (à esquerda) e BURS, TEMPO = 1min03s, ERRO = 0.43% (à direita). Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $f(t)$.

3.1.3 Função Polinomial

$f(t) = t^3 + t^2 + 2, T_1 = -3, T_2 = 3.$

Fizemos neste exemplo os mesmos testes que o exemplo anterior, ou seja, variamos os valores de m e n. A título de comparação, resolvemos cada caso utilizando gradientes conjugados (CG), com o máximo de 40 iterações e precisão de 0.1%. Em todos os casos ele atingiu o máximo de iterações sem atingir a precisão desejada. Vamos exibir apenas os casos $m = 64$, $m = 512$ e $m = 4096$.

O algoritmo BURS forneceu bons resultados em todos os casos, enquanto que pelos outros dois métodos os resultados não foram satisfatórios. Até m e n da ordem de centenas, o algoritmo URS mostrou-se o mais rápido de todos os outros, seguido pelo CG, sendo o mais lento o algoritmo BURS. Nos casos onde m e n eram maiores, o BURS foi o mais rápido, sendo nestes casos o URS o mais lento, comprovando mais uma vez os resultados fornecidos em Rosenfeld (1998). Apesar do algoritmo BURS ser mais lento que os demais para dimensões pequenas, podemos dizer que ele é mais viável em todos os casos, levando-se em conta a melhor qualidade dos resultados apresentados. Este exemplo comprova a eficiência do algoritmo BURS em problemas em que o algoritmo URS não fornece resultados satisfatórios mesmo em dimensões pequenas. Estes resultados podem ser vistos nas Figuras 3.9 a 3.11.

3.1.4 Sismograma

Com o objetivo de mostrar o potencial da interpolação em problemas sísmicos, e como último exemplo unidimensional, reamostramos uma seção sísmica cujos traços não estavam igualmente espaçados (Ver Figura 3.12). Isto se deve ao fato dos receptores não estarem igualmente espaçados, talvez por razões geográficas (correnteza no caso marítimo ou irregularidades no solo) ou queima de algum receptor. Os algoritmos URS, BURS e rBURS foram aplicados para simular a seção sísmica para receptores igualmente espaçados.

O tempo foi discretizado entre 0s e 1.5s com espaçamento de 2ms e a reamostragem uniforme foi feita da seguinte forma: para cada valor de t (tempo) fixo, tomamos os valores que tínhamos para cada posição do receptor e reamostramos nas posições igualmente espaçadas desejadas. Depois disto, para cada posição do receptor fixa, tínhamos os valores para cada valor de t, formando assim um traço. Desta maneira estava feita a reamostragem uniforme. Como este exemplo era muito instável, o algoritmo BURS não apresentou bons resultados, sendo estes inclusive piores que os resultados apresentados pelo URS, fazendo-se necessária a utilização do algoritmo rBURS. Este algoritmo foi aplicado para dois valores diferentes do parâmetro ρ,

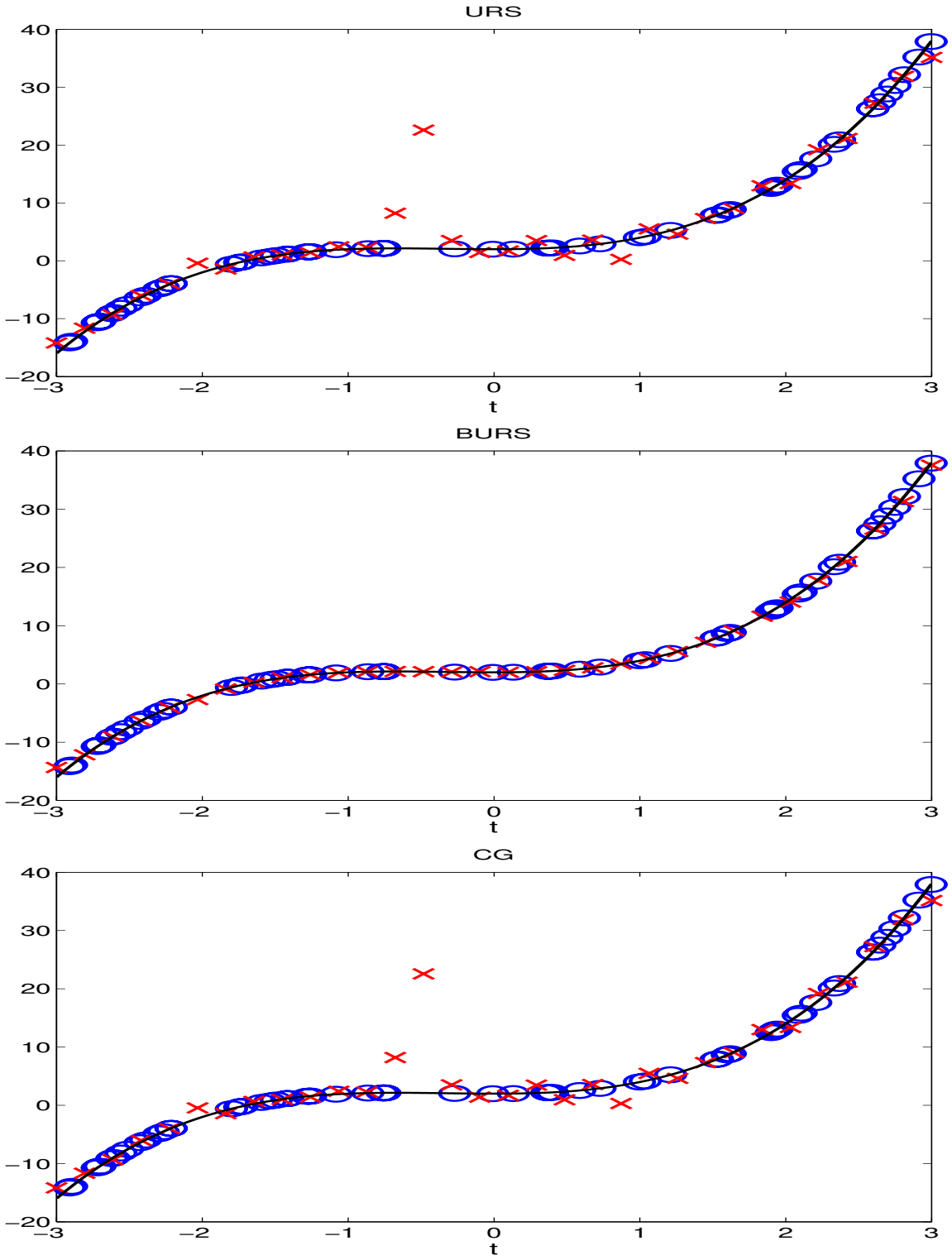


Figura 3.9: Reamostragem uniforme do sinal $f(t) = t^3 + t^2 + 2$, $m = 64$, $n = 32$: URS, TEMPO = 0.001s, ERRO = 31.14% (acima); BURS, TEMPO = 0.047s, ERRO = 3.08% (ao centro) e CG, TEMPO = 0.015s, ERRO = 31.06% (abaixo). Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $f(t)$.

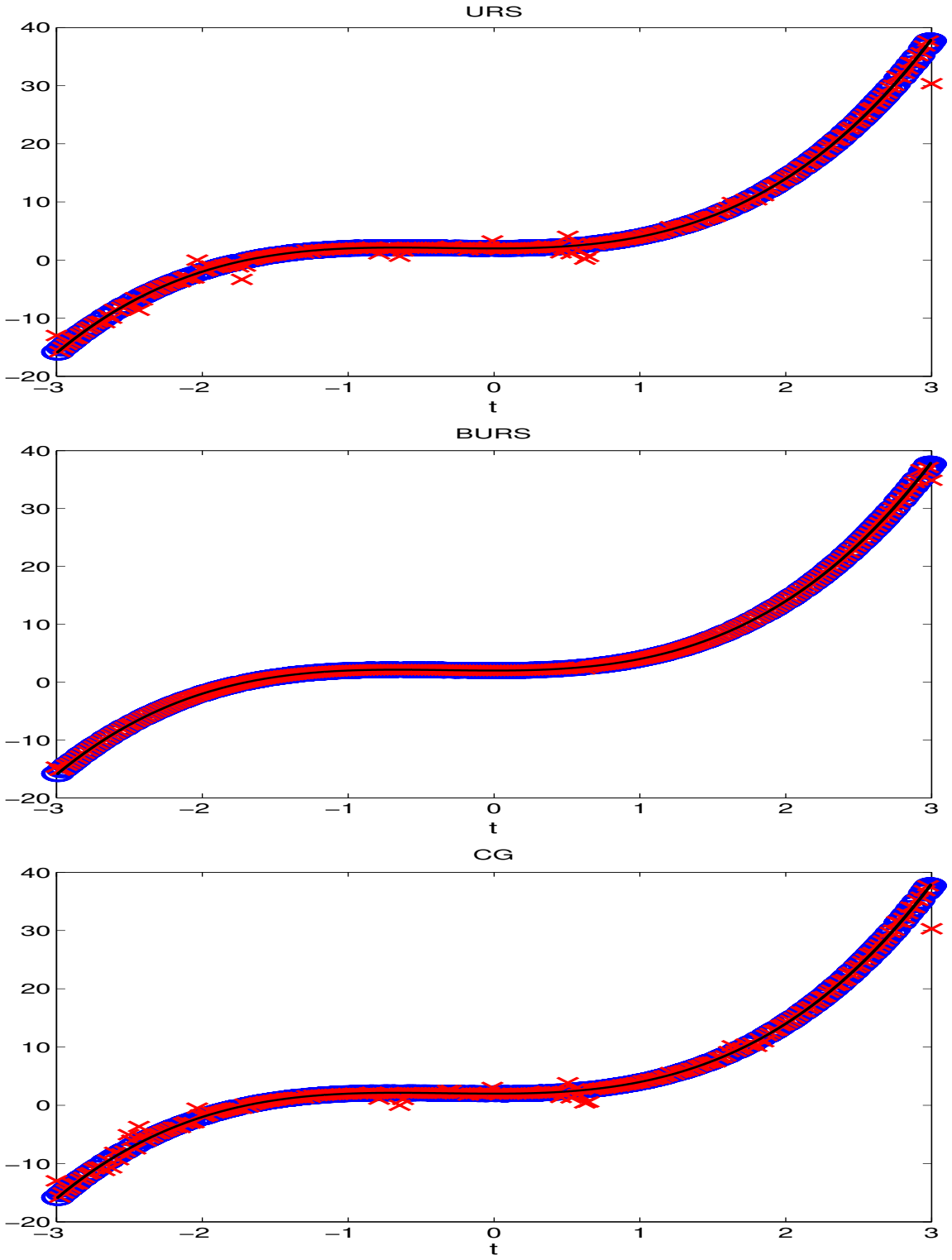


Figura 3.10: Reamostragem uniforme do sinal $f(t) = t^3 + t^2 + 2$, $m = 512$, $n = 256$: URS, TEMPO = 0.235s, ERRO = 05.82% (acima); BURS, TEMPO = 1.093s, ERRO = 01.89% (ao centro) e CG, TEMPO = 0.750s, ERRO = 5.90% (abaixo). Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $f(t)$.

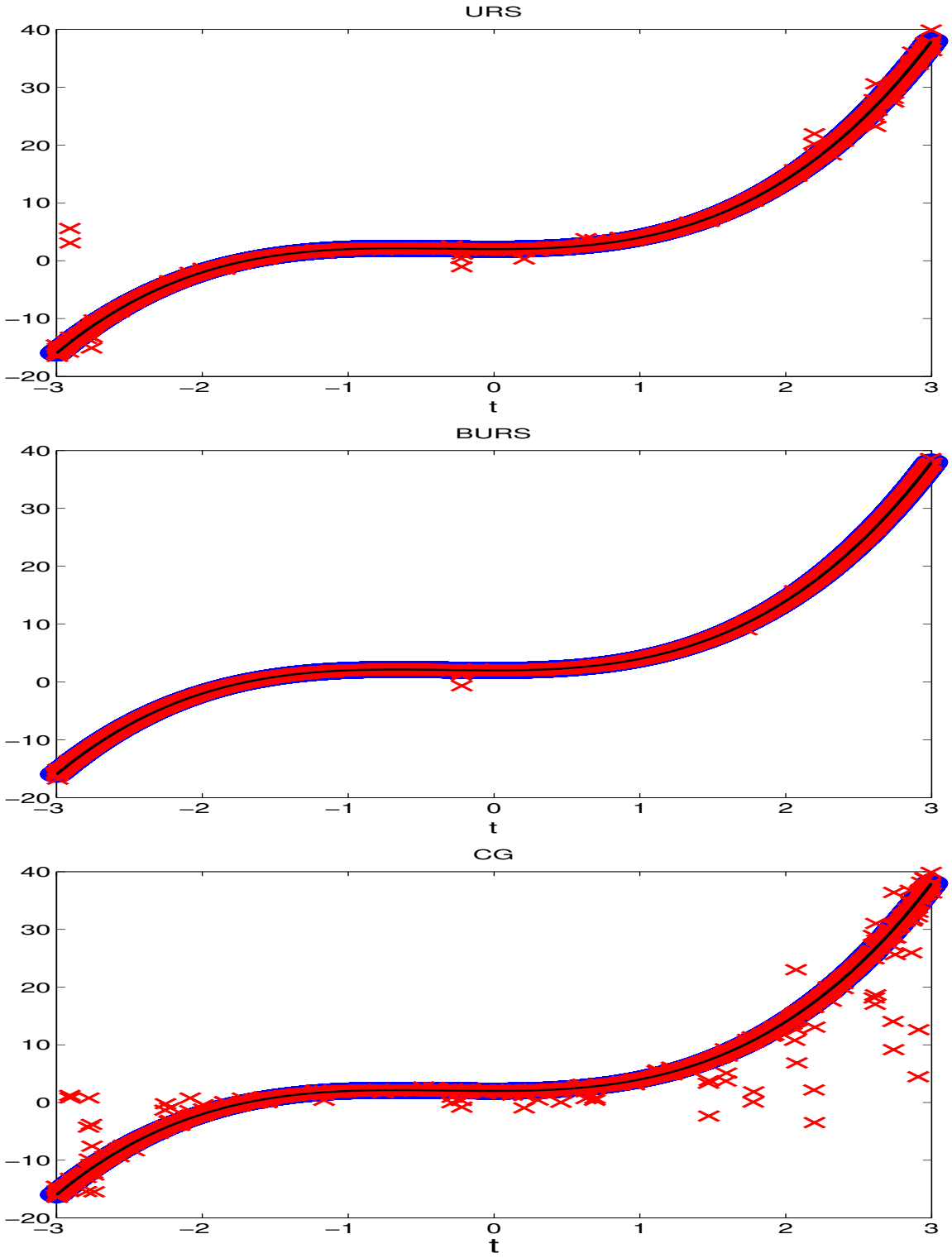


Figura 3.11: Reamostragem uniforme do sinal $f(t) = t^3 + t^2 + 2$, $m = 4096$, $n = 2048$: URS, TEMPO = 2min26s, ERRO = 5.48% (acima); BURS, TEMPO = 1min01s, ERRO = 1.73% (ao centro) e CG, TEMPO = 1min14s, ERRO = 33.01% (abaixo). Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $f(t)$.

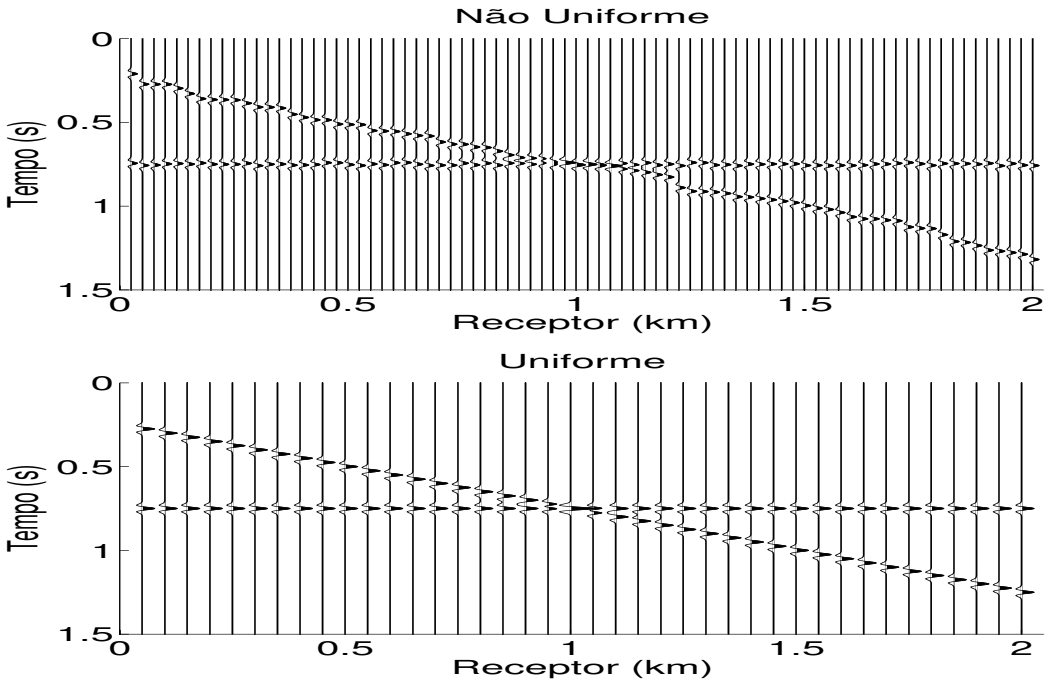


Figura 3.12: Seção sísmica modelada para receptores não igualmente espaçados (acima) e para receptores igualmente espaçados (abaixo).

0.10 e 0.40. Pudemos notar que ele realmente aumentou a estabilidade com relação à solução apresentada pelo BURS, porém com amplitudes menores do que a solução correta. Notamos ainda que estas características são maiores para o maior valor de ρ utilizado. Ressaltamos que o rBURS apresentou também melhores resultados quando comparado ao URS. Os resultados podem ser vistos nas Figura 3.13 e 3.14. Este exemplo comprova a ineficiência do algoritmo BURS para problemas instáveis e exemplifica a teoria vista anteriormente sobre regularização.

3.1.5 Análise do Condicionamento da matriz A

$f(t) = 0.5t^5 - 3t^3 + t^2 + 5, T_1 = -3, T_2 = 3.$
$g(t) = \cos 3t \,\text{sen}\, t + 2\text{sen}\, t, T_1 = -3, T_2 = 3.$

O objetivo deste exemplo é analisar como o número de condição da matriz envolvida no problema de reamostragem uniforme influencia nos resultados de cada método. O número de condição de uma matriz é dado por $K(A) = \|A\|\|A^{-1}\|$ e quanto maior o valor de $K(A)$, pior o condicionamento de A. Aqui adotamos norma-2, assim, $K(A) = \sigma_{max}/\sigma_{min}$, i.e., o número de condição é dado pela divisão do maior valor singular de A, pelo menor. Assim, se $\sigma_{max} \gg \sigma_{min}$, a matriz A é mal condicionada e, no caso de $\sigma_{min} = 0$, $K(A) = \infty$ (Trefethen and Bau, 1997).

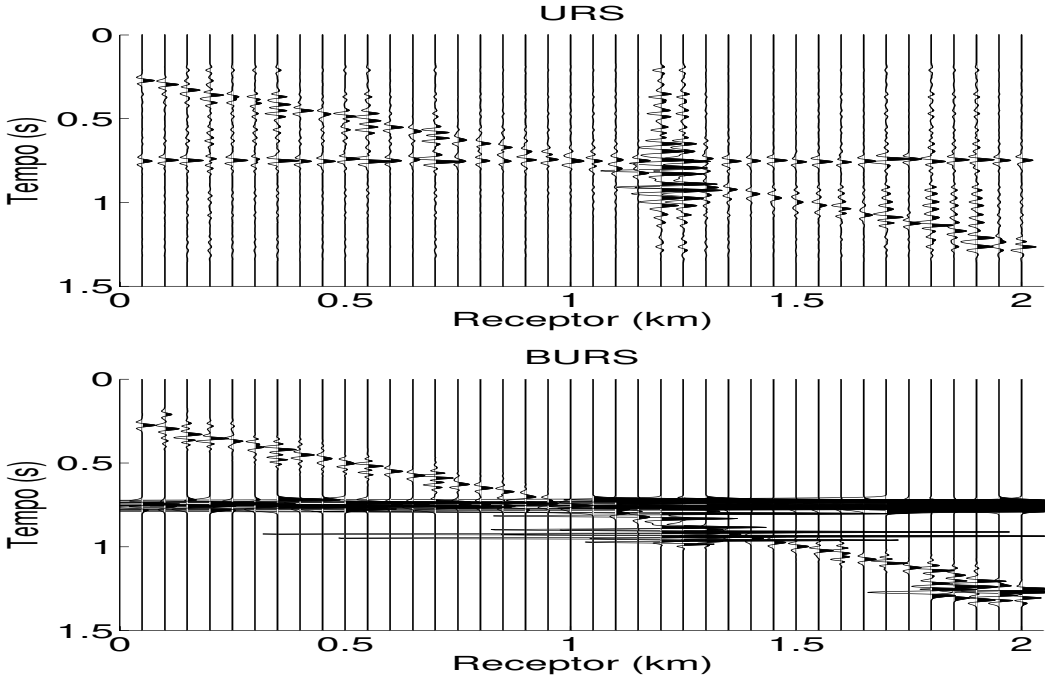


Figura 3.13: Seção sísmica interpolada usando URS (acima) e BURS (abaixo).

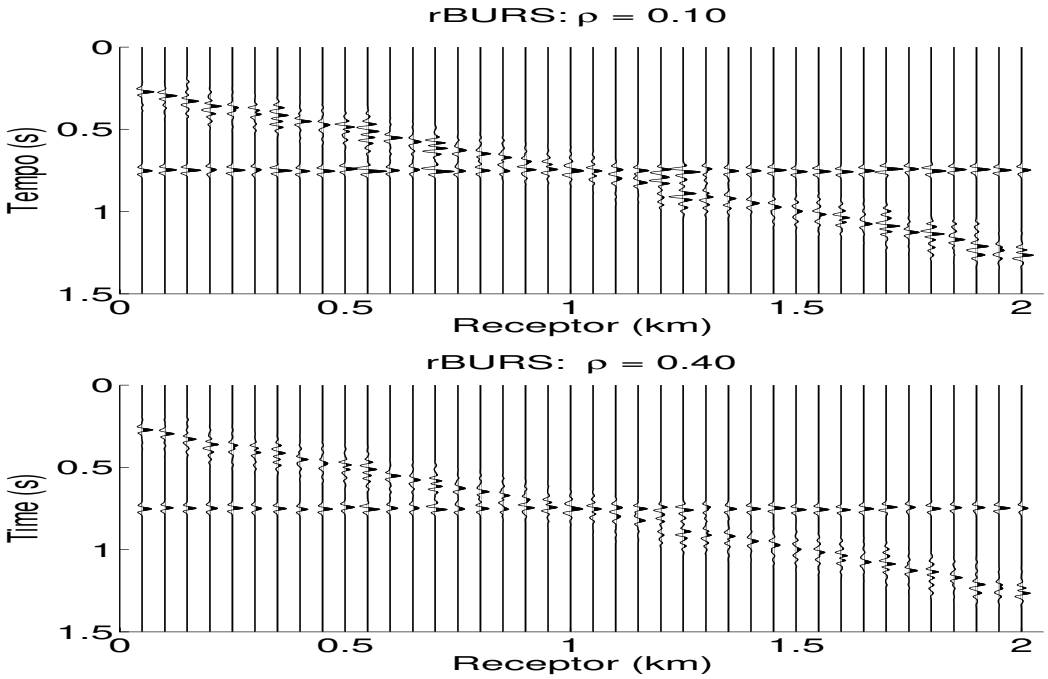


Figura 3.14: Seção sísmica interpolada usando rBURS com $\rho = 0.10$ (acima) e $\rho = 0.40$ (abaixo).

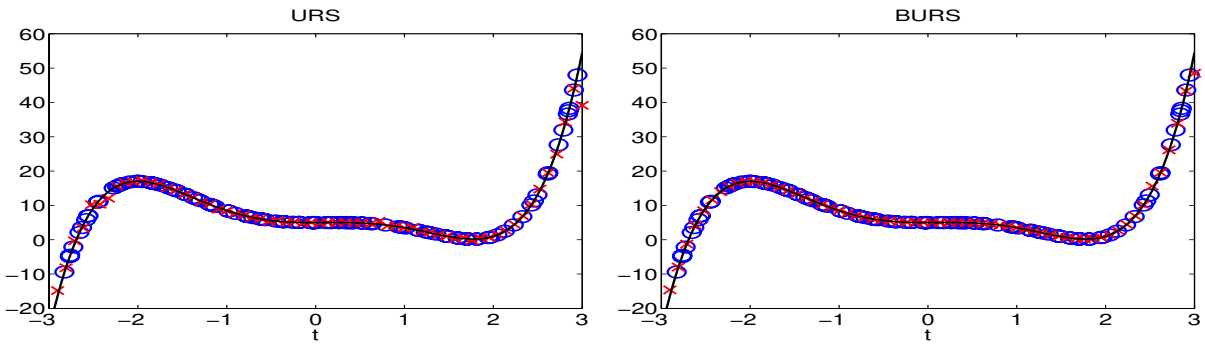


Figura 3.15: Reamostragem uniforme do sinal $f(t) = 0.5t^5 - 3t^3 + t^2 + 5$, $m = 128$, $n = 64$, $K(A) = 10.1$: URS (à esquerda) e BURS (à direita). Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $f(t)$.

Geramos amostras das funções polinomial e trigonométrica acima descritas, em 128 pontos gerados aleatoriamente no intervalo $[T_1, T_2]$, e reamostramos tais funções em 64 pontos igualmente espaçados, utilizando os algoritmos URS e BURS. Para $f(t)$ e $g(t)$, a matriz do sistema linear era a mesma, uma vez que os pontos t_i's e τ_j's eram os mesmos. Fizemos isto duas vezes, sendo que no primeiro caso, a matriz A tinha número de condição 10.1 e no segundo 476.6. A remostragem da função trigonométrica $g(t)$ apresentou bons resultados pelos dois métodos, nos dois casos. Já com relação à função polinomial $f(t)$, os resultados fornecidos pelo URS no segundo caso não foram satisfatórios. Analisamos também os números de condição das submatrizes de A utilizadas no BURS. No primeiro caso, estes valores variaram entre 1.03×10^4 e 1.38×10^{12} e no segundo caso entre 1.86×10^2 e 1.19×10^{11}. Notamos que, apesar destes números serem muito grandes, os resultados fornecidos pelo BURS em todos os casos, foi muito bom. Acreditamos que o fato do algoritmo URS apresentar bons resultados para os dois valores de $K(A)$ apenas para a função trigonométrica está ligado ao fato desta função ser banda-limitada. Todos os resultados podem ser vistos nas Figuras 3.15 a 3.18.

3.2 Caso Bidimensional

3.2.1 Função Trigonométrica

Como primeiro exemplo bidimensional tomamos a função $f(x, y) = 3\text{sen}x + 5\cos 2y$, amostrada no intervalo $[-1, 1] \times [-1, 1]$ em 1024 pontos Cartesianos, gerados aleatoriamente a partir de uma distribuição uniforme, e reamostramos tal função em 256 pontos de uma grade Cartesiana usando os algoritmos URS e BURS (ver Figura 3.19). Como este é outro exemplo de

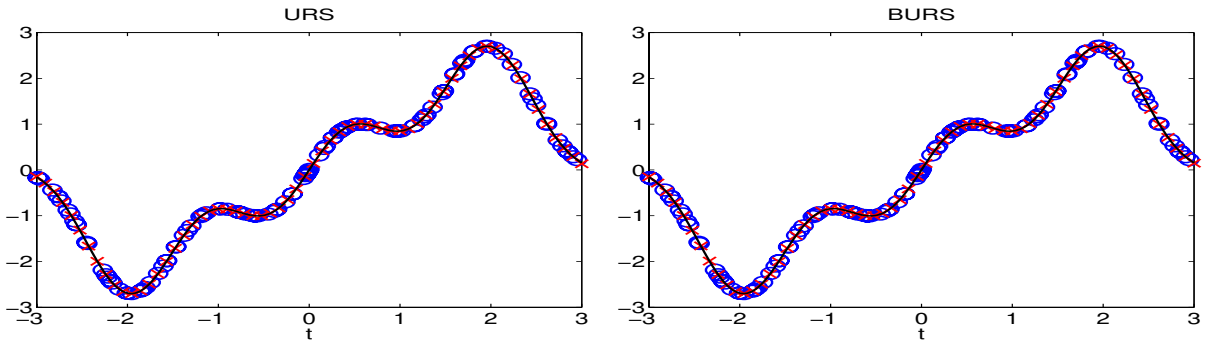


Figura 3.16: Reamostragem uniforme do sinal $g(t) = \cos 3t \operatorname{sen} t + 2\operatorname{sen} t$, $m = 128$, $n = 64$, $K(A) = 10.1$: URS (à esquerda) e BURS (à direita). Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $g(t)$.

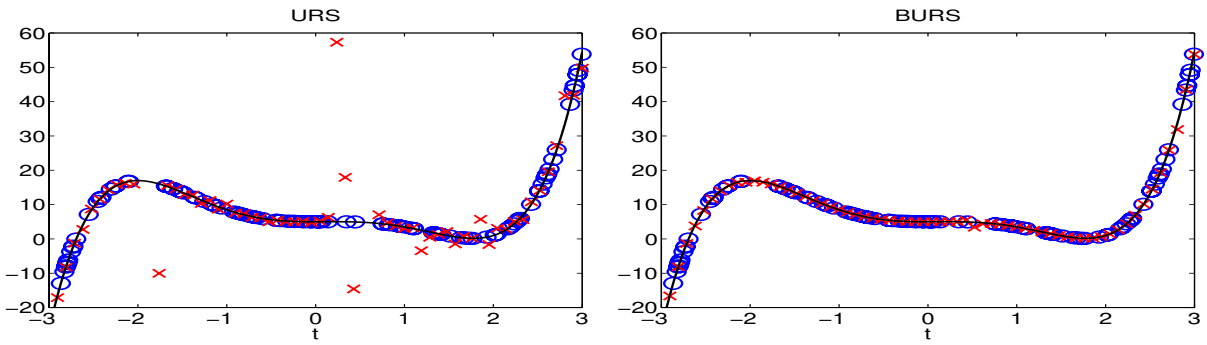


Figura 3.17: Reamostragem uniforme do sinal $f(t) = 0.5t^5 - 3t^3 + t^2 + 5$, $m = 128$, $n = 64$, $K(A) = 476.6$: URS (à esquerda) e BURS (à direita). Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $f(t)$.

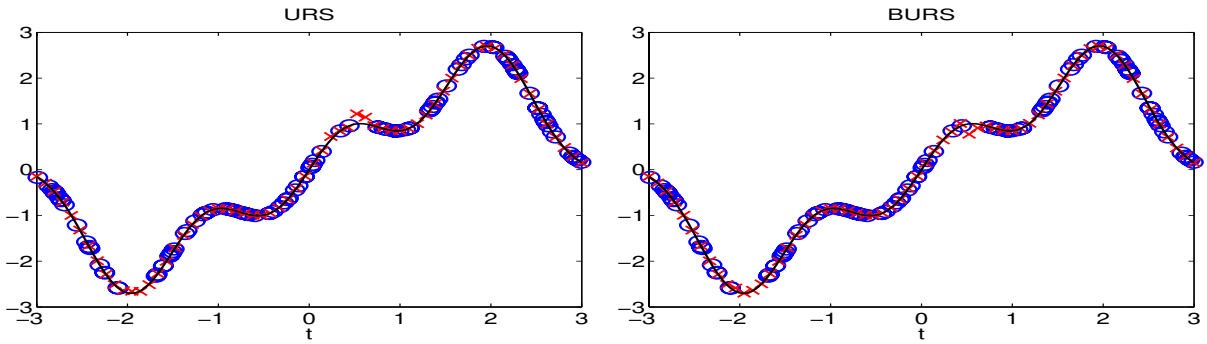


Figura 3.18: Reamostragem uniforme do sinal $g(t) = \cos 3t \operatorname{sen} t + 2\operatorname{sen} t$, $m = 128$, $n = 64$, $K(A) = 476.6$: URS (à esquerda) e BURS (à direita). Os símbolos $\circ$'s correspondem às amostras, os $\times$'s representam a reamostragem uniforme e o traço contínuo representa o sinal $g(t)$.

uma combinação de senos e cossenos (banda limitada), os dois algoritmos apresentaram bons resultados, porém, os fornecidos pelo BURS foram melhores. Como as dimensões do problema não são tão grandes, o URS ainda foi praticável, sendo que seu tempo de execução foi quase 10 vezes menor que o do BURS. Os resultados podem ser vistos na Figura 3.20.

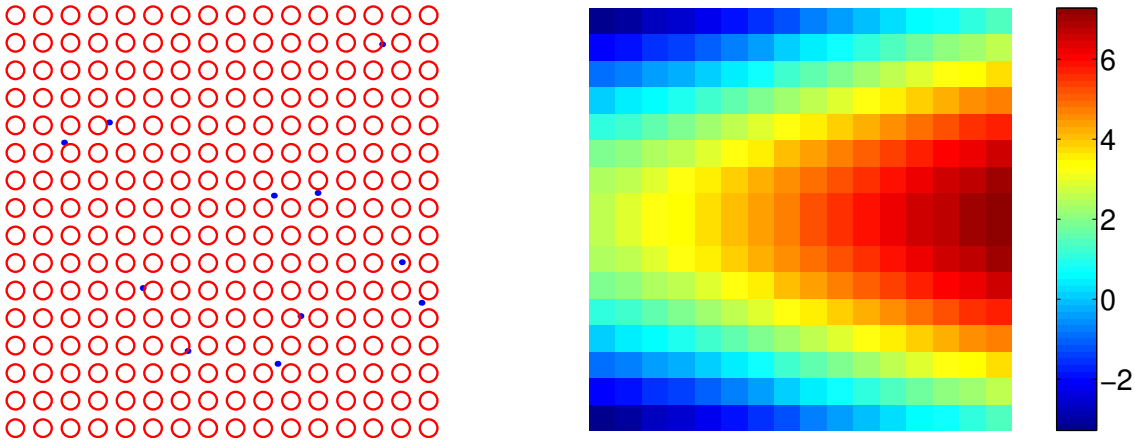


Figura 3.19: Reamostragem uniforme da função $f(x, y) = 3\text{sen}(x) + 5\cos(2y)$ (à esquerda): os símbolos .'s correspondem aos pontos em que a função é conhecida e os ○'s representam os pontos reamostrados. Imagem real da função $f(x, y) = 3\text{sen}(x) + 5\cos(2y)$ nos 256 pontos distribuídos uniformemente no intervalo $[-1, 1] \times [-1, 1]$ (à direita).

3.2.2 Função Não Linear

Neste exemplo, utilizamos a função $f(x, y) = \sqrt{x^2 + y^2}$, amostrada em 1512 pontos, distribuidos radialmente no intervalo $[-1.5, 1.5] \times [-1.5, 1.5]$, e reamostramos tal função em 324 pontos de uma grade Cartesiana usando os algoritmos URS e BURS (ver Figura 3.21). Apenas o algoritmo BURS apresentou resultado satisfatório, já que o algoritmo URS não aproximou bem a imagem nas bordas. Assim como no exemplo anterior, o URS foi mais rápido, sendo que seu tempo de execução foi mais de 20 vezes menor que o do BURS. Os resultados podem ser vistos na Figura 3.22.

3.2.3 Imagem da Lena I

Neste exemplo utilizamos uma imagem da Lena com “buracos", com dimensões 128×128 pixels, obtida de um recorte de uma imagem da Lena 256×256 para que tivesse melhor resolução. A imagem foi corrompida em 5%, 15%, 25%, 35% e 45%, ou seja, em cada caso uma certa porcentagem dos pontos da imagem eram desconhecidos (ver Figura 3.23). Como

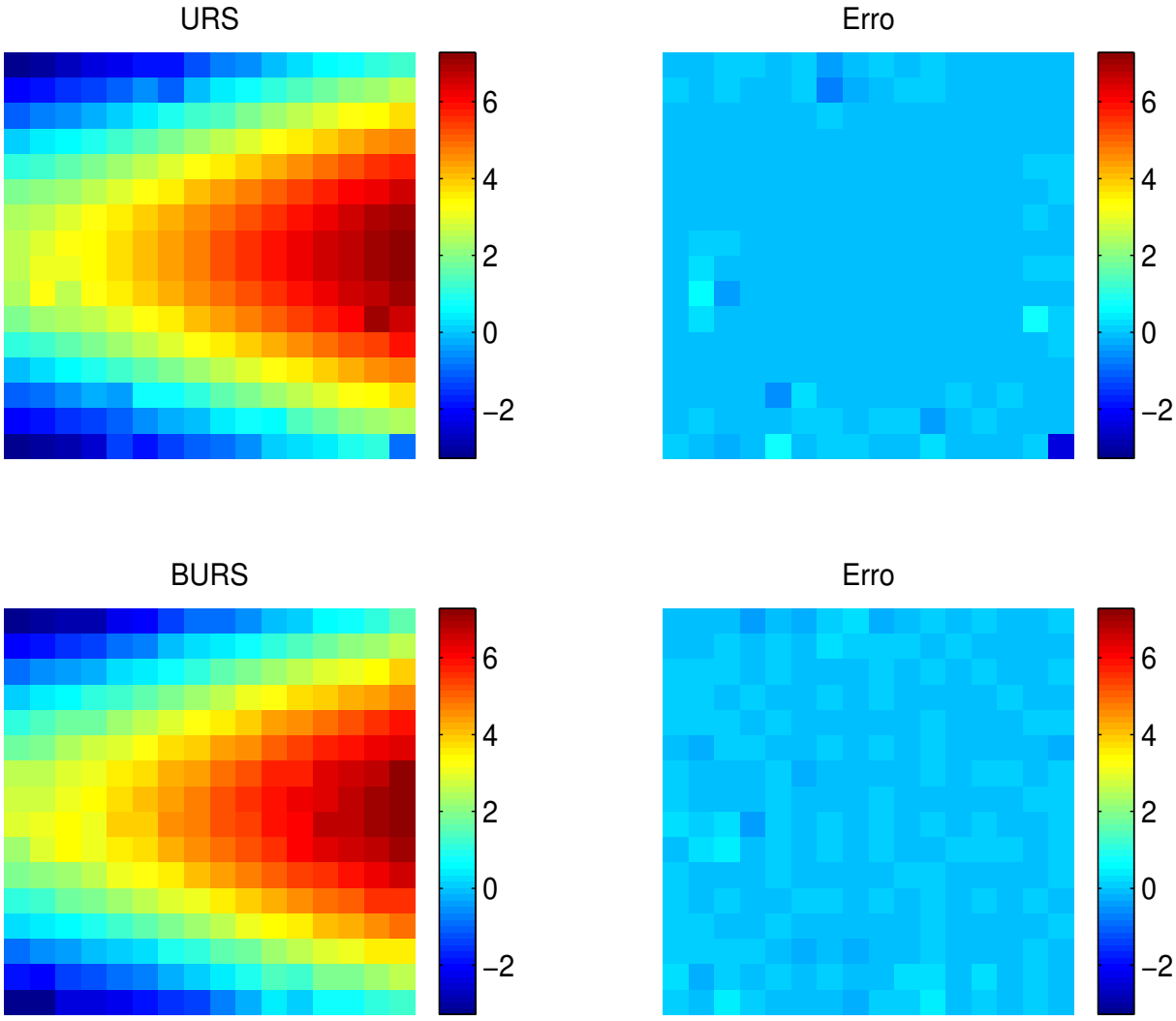


Figura 3.20: Reamostragem uniforme da função $f(x, y) = 3\text{sen}x + 5\cos 2y$: imagem interpolada usando URS, TEMPO = 2.53s (acima) e imagem interpolada usando BURS, TEMPO = 23.438s, (abaixo).

a imagem estava sob a forma de uma matriz, tratamos cada elemento a_{ij} da matriz como o valor de $f(i, j)$. Os “buracos"foram gerados colocando 0 em n pontos da imagem que foram selecionados aleatoriamente. Este é um exemplo em que os pontos que queremos interpolar não estão uniformemente espaçados, ou seja, como trata-se do caso bidimensional, os pontos não estão numa grade Cartesiana, porém são um subconjunto de pontos uniformemente espaçados.

Para recuperação de tal imagem, testamos os algoritmos URS, BURS e CG, com $tol = 0.001$. Como só podemos utilizar os algoritmos para encontrar valores em uma malha uniforme, dividimos o problema em quatro etapas. Em cada etapa consideramos, respectivamente,

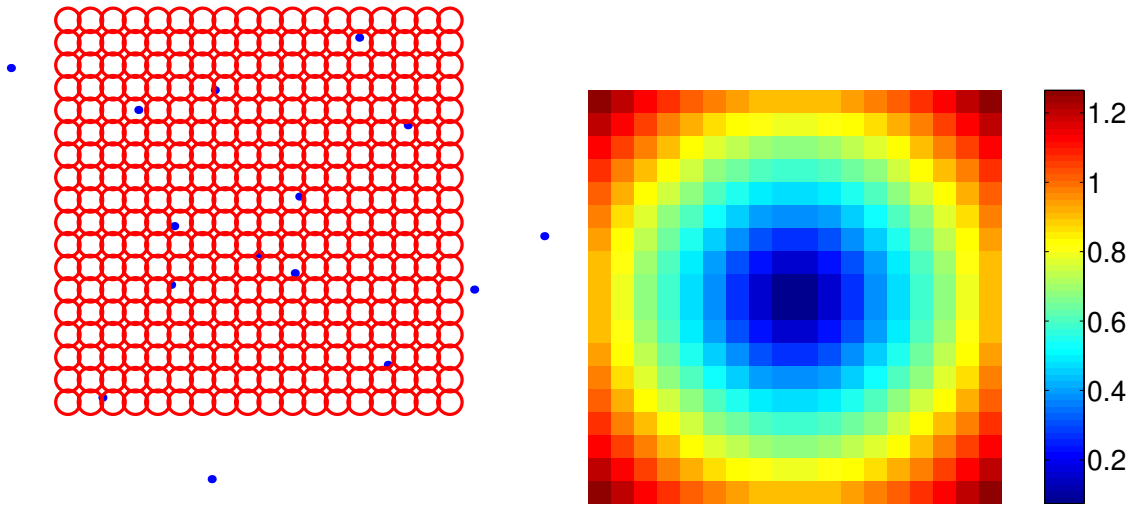


Figura 3.21: Reamostragem uniforme da função $f(x, y) = \sqrt{x^2 + y^2}$ (à esquerda): os símbolos .'s correspondem aos pontos em que a função é conhecida e os ○'s representam os pontos reamostrados. Imagem real da função $f(x, y) = \sqrt{x^2 + y^2}$ nos 324 pontos distribuídos uniformemente no intervalo $[-1, 1] \times [-1, 1]$ (à direita).

os casos onde as coordenadas (x, y) eram: par–par, ímpar–par, par–ímpar e ímpar–ímpar, sendo $\Delta x = 2$ e $\Delta y = 2$. Sobrepusemos à imagem apenas os pontos que faltavam, "jogando fora"o que não precisávamos. O erro relativo em cada caso foi calculado dividindo-se a norma da diferença entre a imagem original e a obtida, pela norma deda imagem original. A norma utilizada foi de Frobenius, que é dada pela raiz quadrada da soma dos quadrados de todos os elementos da matriz. Note que, os pontos calculados (vetor X) não variam com relação à porcentagem de pontos removidos da imagem original; o que muda é o tamanho do vetor b, uma vez que este está relacionado com o número de pontos que conhecemos da imagem, que variou de 95% a 55%. Assim, o tempo de execução dos algoritmos diminui com o aumento da porcentagem de recuperação da imagem. Em todos os casos, o algoritmo CG convergiu, alcançando a precisão desejada. Os algoritmos URS e CG apresentaram resultados muito parecidos, sendo que os erros relativos só tiveram diferença a partir da quarta casa decimal, não considerada aqui. Os resultados fornecidos pelo BURS foram um pouco melhor que os demais na maioria dos casos, sendo seu tempo de execução significativamente menor que os demais em todos ele, e, portanto, podemos dizer que o URS e o CG não são praticáveis para problemas dessa ordem. Os resultados estão apresentados nas Figuras 3.24 a 3.28.

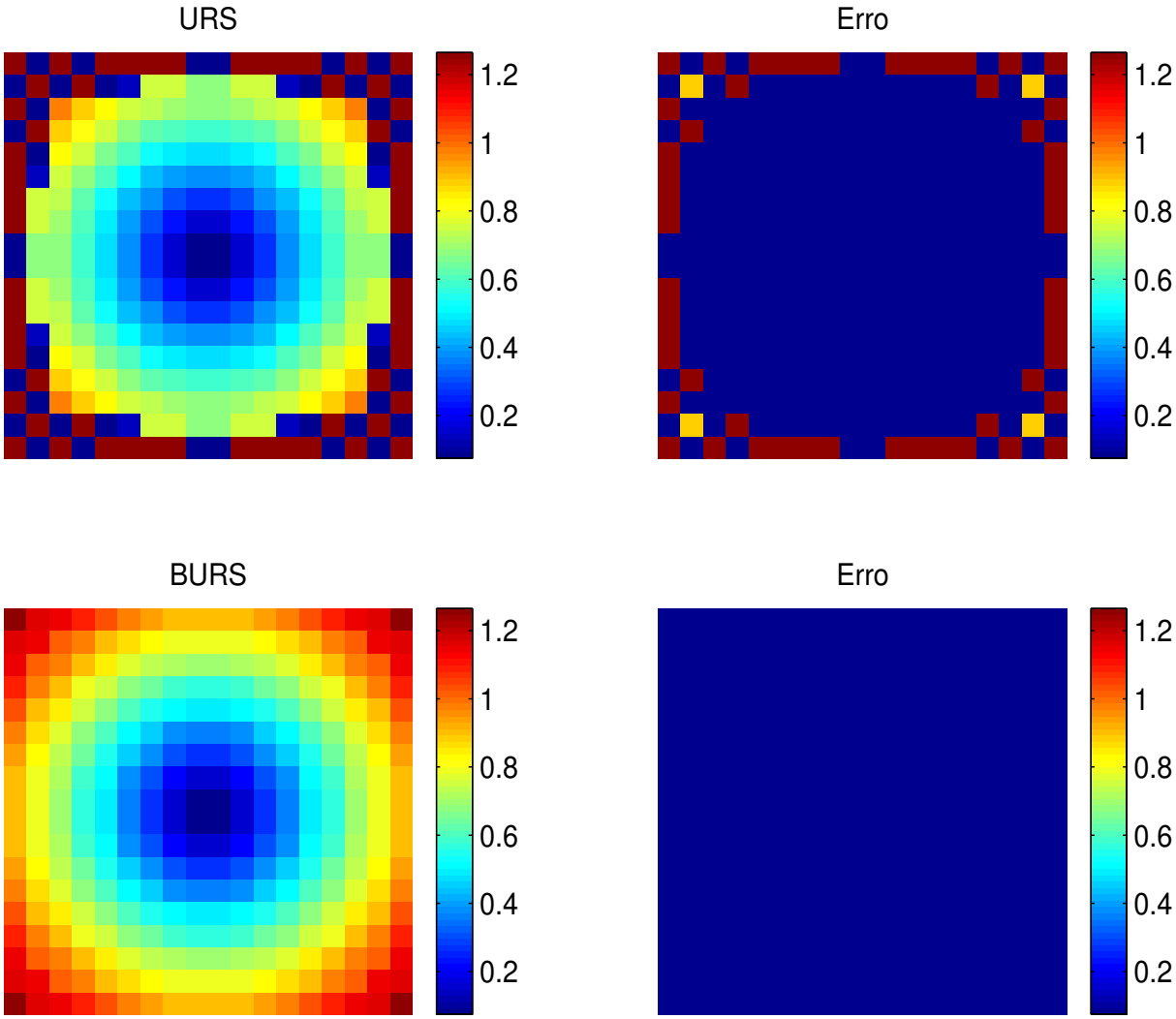


Figura 3.22: Reamostragem uniforme da função $f(x, y) = \sqrt{x^2 + y^2}$: imagem interpolada usando URS, TEMPO = 1.609s (acima) e imagem interpolada usando BURS, TEMPO = 37.562s (abaixo).

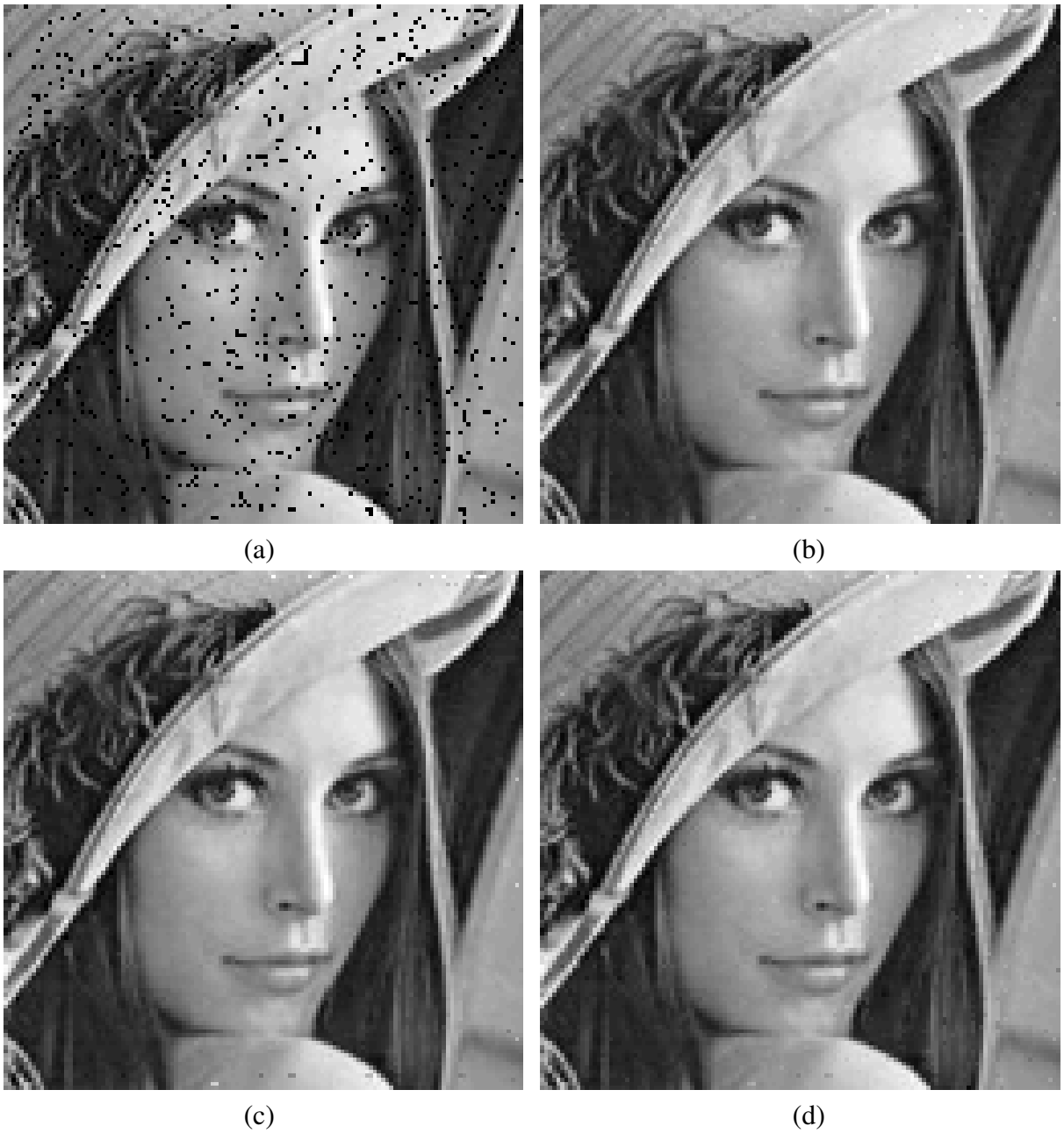
3.2.4 Imagem da Lena II

Neste exemplo, tentamos recuperar um imagem da Lena, com dimensões 95×95 pixels, obtida também de um recorte de uma imagem da Lena 256×256, corrompida numa faixa de dimensões 5×50 pixels. Para isto, utilizamos apenas o algoritmo BURS, já que, como pudemos ver no exemplo anterior, os demais métodos são inviáveis computacionalmente. Infelizmente, constatamos que esse método não se aplica em exemplos deste tipo. Os resultados podem ser vistos na Figura 3.29. Os erros relativos foram calculados como no exemplo anterior.

Figura 3.23: Imagem Original.

3.2.5 Sismograma

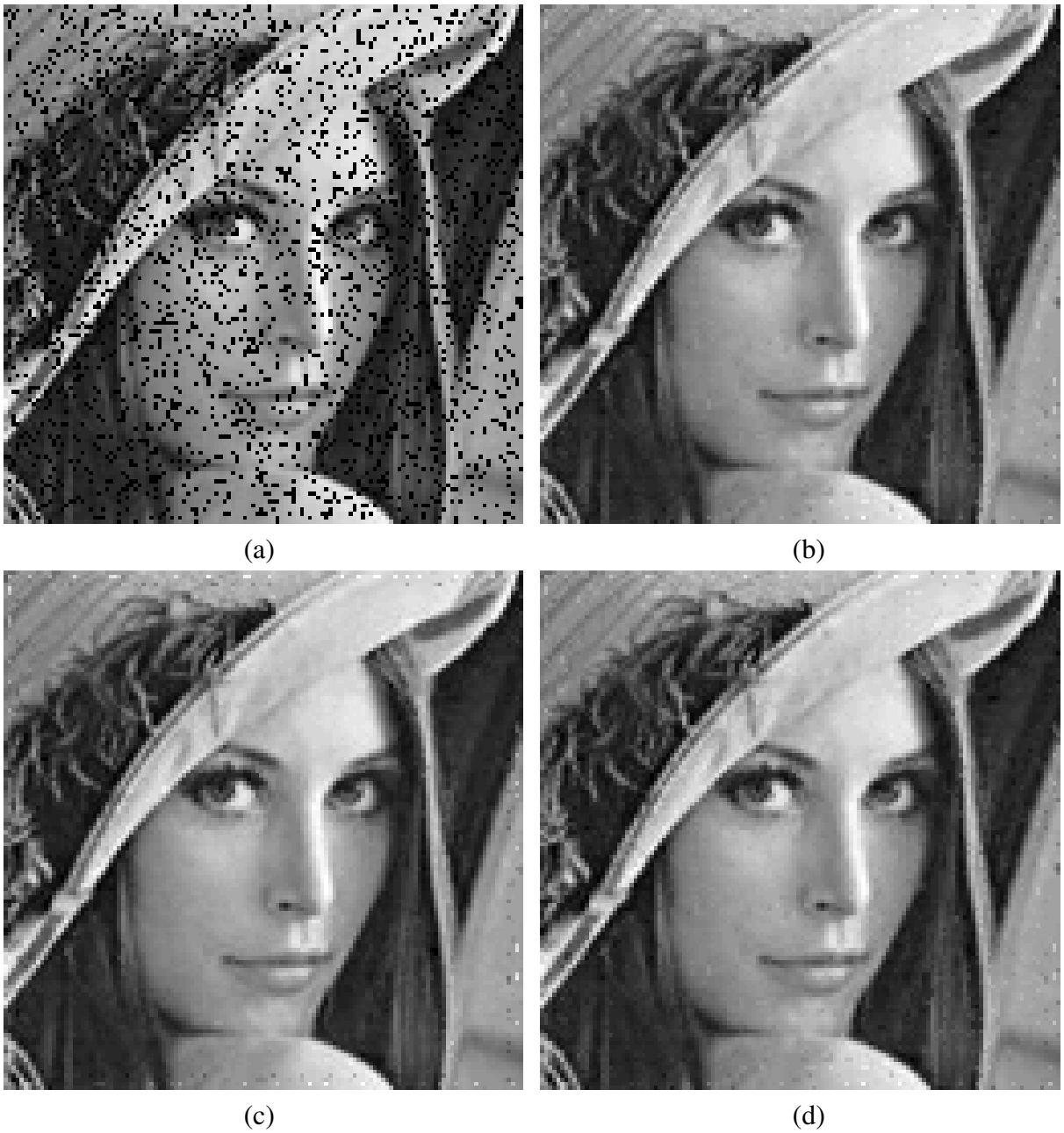
Um outro exemplo de aplicação à geofísica é a reconstrução de imagens. Assim, neste exemplo, trabalhamos com a imagem de uma seção sísmica de 376×40 pixels, como exibido na Figura 3.30. Os dados foram corrompidos, em 30% e 50%, como no Exemplo 3.2.3. Para a reconstrução de tais imagens, utilizamos apenas o algoritmo BURS, devido a viabilidade prática do método. Os resultados apresentados para os casos 30% e 50% podem ser vistos na Figura 3.31. Em ambos os casos os erros relativos, que foram calculados como no exemplo anterior, mostraram-se satisfatórios. Do ponto de vista prático, estes resultados são bons pois permitem uma melhor interpretação da imagem, uma vez que ela possui melhor resolução do que a imagem corrompida.



(a) (b)

(c) (d)

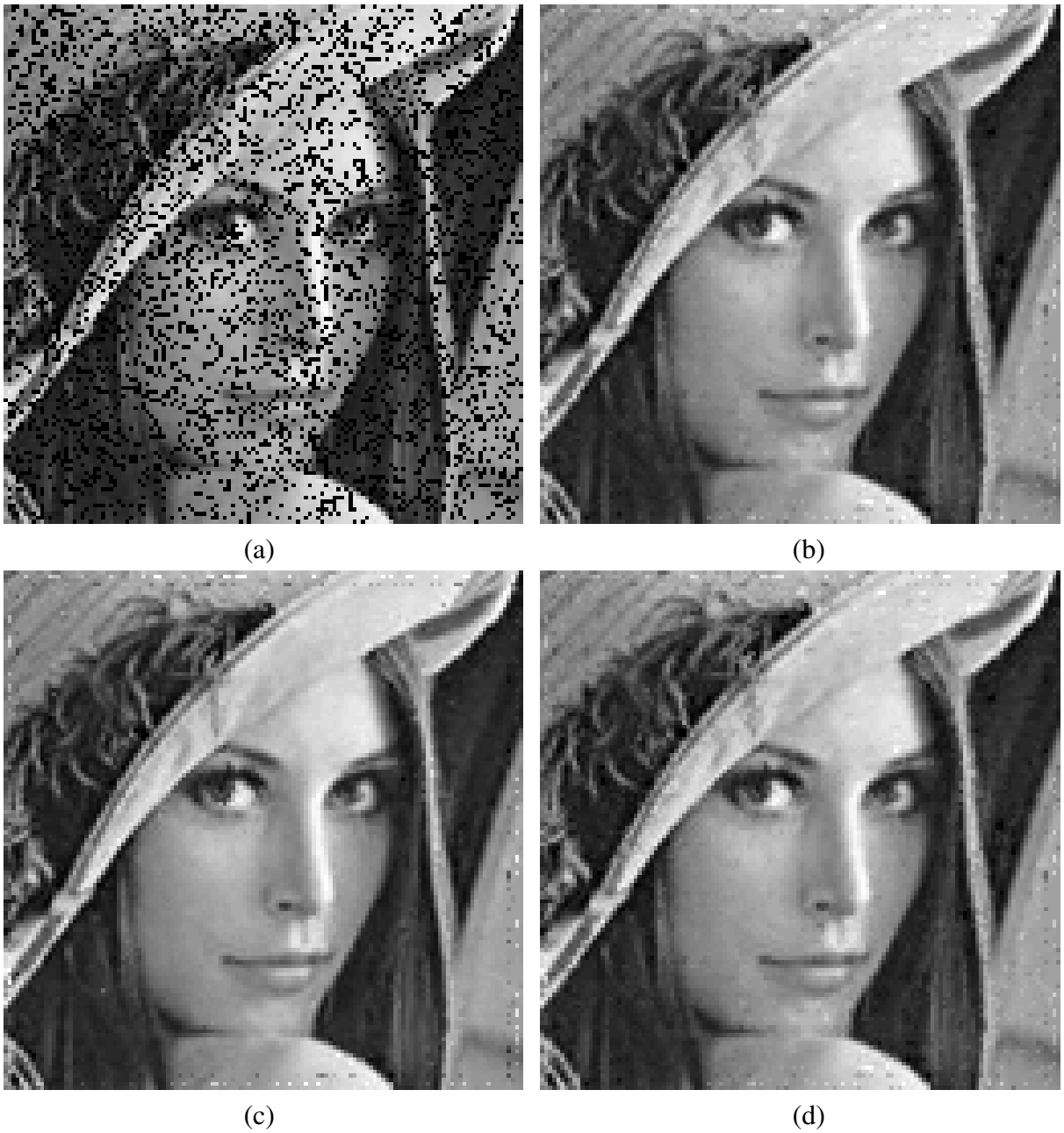
Figura 3.24: (a) Imagem corrompida 05%. (b) Imagem recuperada por URS, TEMPO = 27h21min e ERRO = 2.28%. (c) Imagem recuperada por BURS, TEMPO = 14min52s e ERRO = 2.16%. (d) Imagem recuperada utilizando Gradientes Conjugados, TEMPO = 24h30min e ERRO = 2.28%.



(a) (b)

(c) (d)

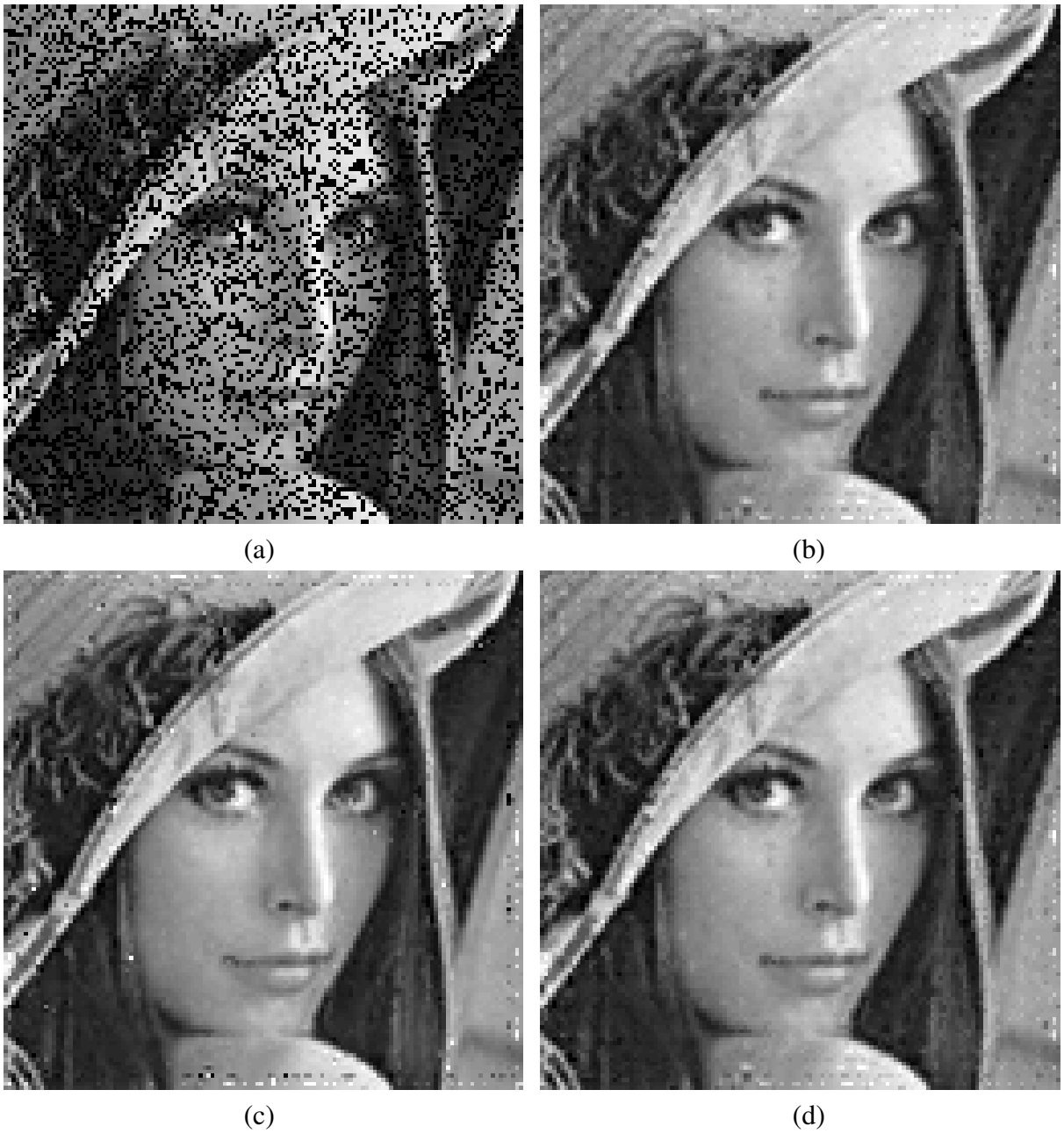
Figura 3.25: (a) Imagem corrompida 15%. (b) Imagem recuperada por URS, `TEMPO` = 21h26min e `ERRO` = 4.02%. (c) Imagem recuperada por BURS, `TEMPO` = 11min34s e `ERRO` = 3.81%. (d) Imagem recuperada utilizando Gradientes Conjugados, `TEMPO` = 19h52min e `ERRO` = 4.02%.



(a) (b)

(c) (d)

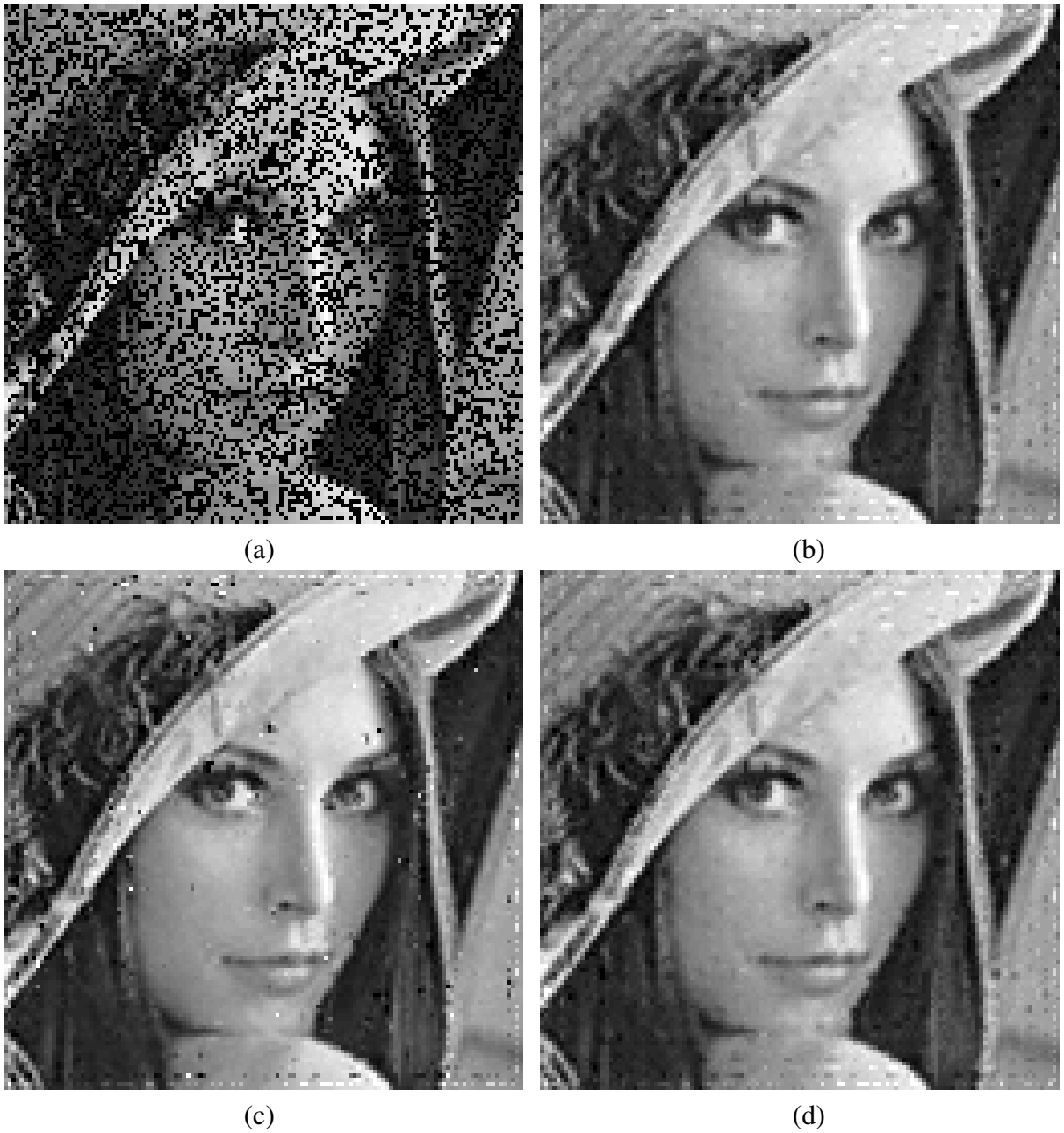
Figura 3.26: (a) Imagem corrompida 25%. (b) Imagem recuperada por URS, TEMPO = 16h57min e ERRO = 5.28%. (c) Imagem recuperada por BURS, TEMPO = 10min36s e ERRO = 5.22%. (d) Imagem recuperada utilizando Gradientes Conjugados, TEMPO = 16h12min e ERRO = 5.28%.



(a) (b)

(c) (d)

Figura 3.27: (a) Imagem corrompida 35%. (b) Imagem recuperada por URS, `TEMPO` = 13h55min e `ERRO` = 6.36%. (c) Imagem recuperada por BURS, `TEMPO` = 9min54s e `ERRO` = 11.63%. (d) Imagem recuperada utilizando Gradientes Conjugados, `TEMPO` = 13h15min e `ERRO` = 6.36%.



(a) (b)

(c) (d)

Figura 3.28: (a) Imagem corrompida 45%. (b) Imagem recuperada por URS, TEMPO = 11h19min e ERRO = 7.51%. (c) Imagem recuperada por BURS, TEMPO = 9min16s e ERRO = 13.16%. (d) Imagem recuperada utilizando Gradientes Conjugados, TEMPO = 10h53min e ERRO = 7.51%.

(a) (b)

Figura 3.29: (a) Imagem corrompida. (b) Imagem recuperada por BURS, TEMPO = 5min39s e ERRO = 13.71%.

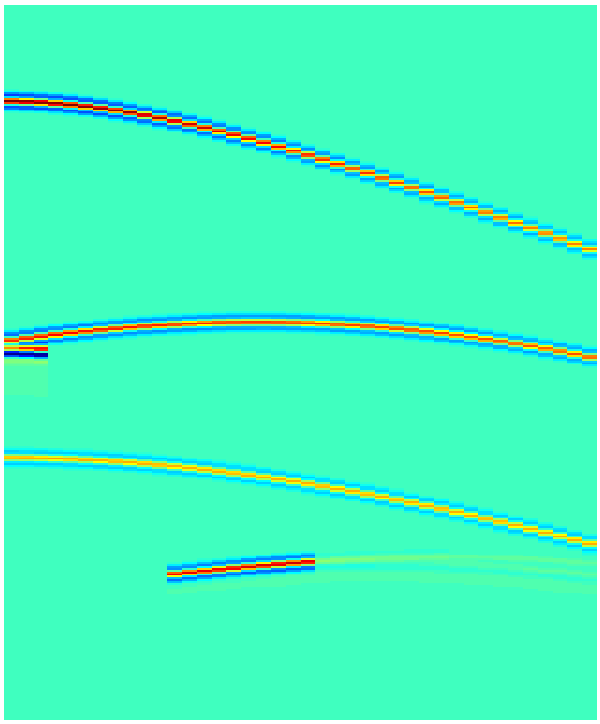


Figura 3.30: Imagem Original.

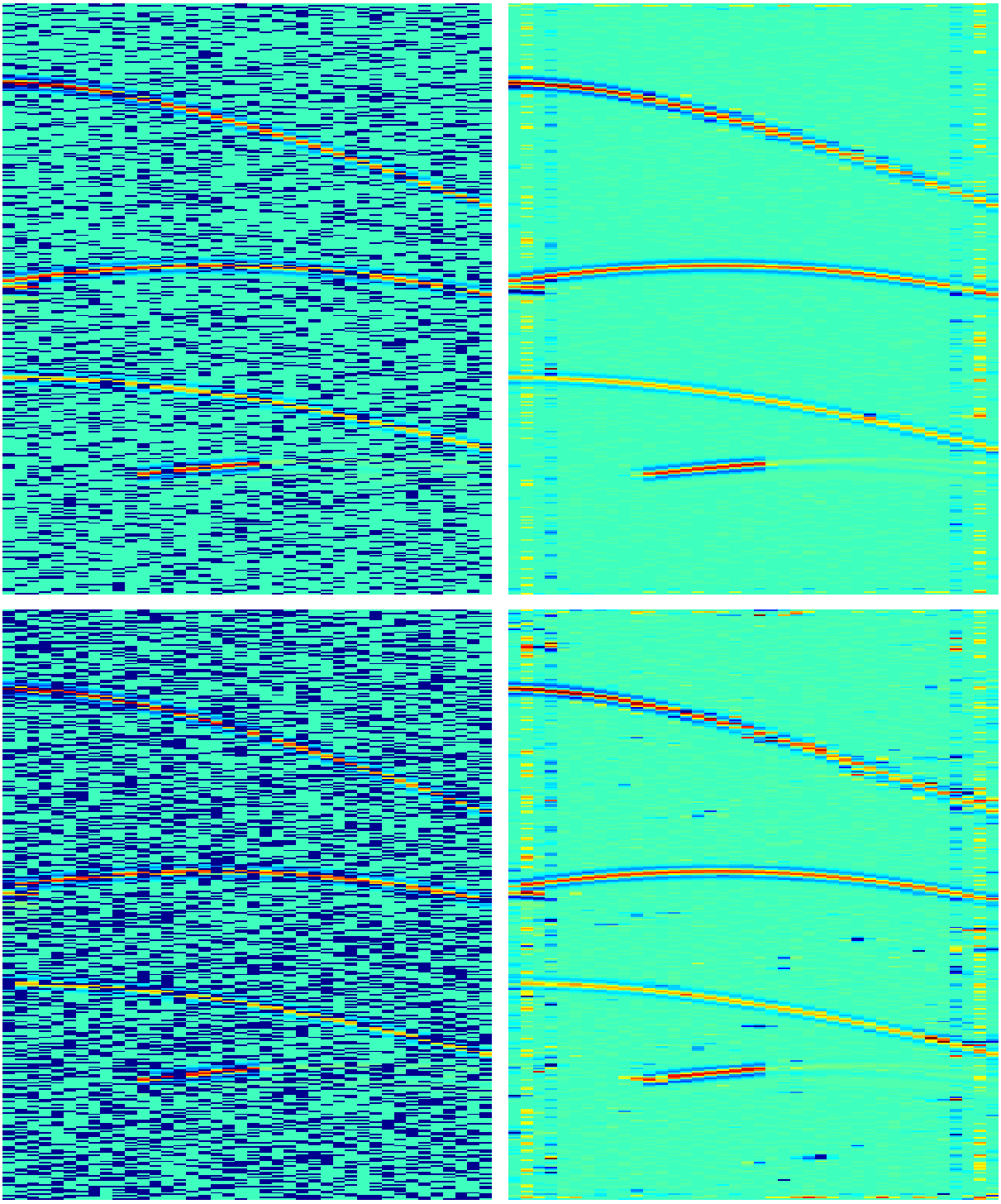


Figura 3.31: Imagem corrompida 30% (acima à esquerda). Imagem recuperada por BURS, TEMPO = 8min16s e ERRO = 6.61% (acima à direita). Imagem corrompida 50% (abaixo à esquerda). Imagem recuperada por BURS, TEMPO = 7min03s e ERRO = 29.88% (abaixo à direita).

Conclusão

Neste trabalho, estudamos o problema de reamostragem uniforme e vimos que, utilizando o Teorema de Shannon (Capítulo 1), este pode ser formulado como um problema de se resolver um sistema de equações lineares. Com o objetivo de constatar a utilidade da reamostragem uniforme, estudamos a Transformada de Fourier Discreta (DFT) e comparamos, através de um exemplo, as aproximações para a Transformada de Fourier de um certo sinal, fornecidas pela DFT, utilizando primeiramente uma amostra não uniforme do sinal e, em seguida, uma reamostragem uniforme da amostra original. Verificamos que, o resultado fornecido pela última amostra é bem melhor que o fornecido pela primeira, justificando assim a necessidade de se reamostrar os dados não uniformes (Capítulo 2).

Em seguida, estudamos métodos para resolver o sistema linear em questão. Primeiramente, consideramos a solução da pseudoinversa (SVD), em um processo denominado URS. Este método é ótimo no sentido de quadrados mínimos de norma mínima, porém, em problemas onde são envolvidas muitas equações e variáveis (da ordem de milhares) ele é impraticável. Estudamos então o algoritmo BURS, complemento sub-ótimo do algoritmo URS, o qual é eficiente e praticável para problemas grandes e outras situações. Neste método, apenas um número limitado de valores são usados para gerar cada ponto da grade uniforme. Com esses valores é construído um conjunto de equações lineares apropriado, que é resolvido utilizando a SVD. Apesar de ser eficiente na maioria das vezes, em problemas instáveis, o BURS não fornece bons resultados. Para contornar este problema, introduzimos a algoritmo rBURS, versão regularizada do BURS, eficiente para problemas deste tipo. Por último, com o objetivo de comparação, também estudamos o método dos Gradientes Conjugados.

Primeiramente, formulamos alguns exemplos no caso unidimensional, com o objetivo de testar os métodos discutidos anteriormente. Consideramos dois tipos diferentes de funções e variamos o tamanho das amostras, de 64 a 4096 pontos, sendo estas reamostradas sempre numa quantidade de pontos igual à metade da quantidade dos pontos originais. Em todos casos, comparamos os resultados dos algoritmos URS e BURS, e em alguns deles, também aplicamos o algoritmo CG. Notamos que em problemas até a ordem de centenas, o BURS é o mais lento

dos métodos, apesar de apresentar bons resultados em todos os casos, enquanto o URS apresentou resultados satisfatórios apenas nos exemplos onde a função era um combinação de senos e cossenos. Nos outros exemplos, o método CG também foi testado, apresentando resultados semelhantes ao URS e sendo mais lento que este em problemas até a ordem de centenas. Em problemas maiores, o BURS foi o mais rápido dos métodos em todos os exemplos considerados, seguido pelo CG, sendo o mais lento nestes casos o algoritmo URS.

Como exemplo de aplicação à geofísica, ainda no caso unidimensional, aplicamos os algoritmos URS e BURS na reamostragem de um sismograma, para simular uma seção sísmica com receptores igualmente espaçados. Verificamos que nenhum dos dois algoritmos apresentaram bons resultados, devido à instabilidade do problema. Aplicamos então o algoritmo rBURS, com dois parâmetros de regularização diferentes, e obtevemos resultados satisfatórios, que aproximaram bem a seção sísmica em questão.

No caso bidimensional, começamos reamostrando dois tipos de funções. No primeiro caso, conhecíamos a função, uma combinação de senos e cossenos (banda limitada), em pontos cartesianos que haviam sido gerados aleatoriamente num intervalo pré-determinado. Os dois algoritmos, URS e BURS, apresentaram bons resultados, devido à maneira que os pontos estavam distribuídos e ao tipo de função. No segundo caso, tomamos outra função, cujas amostras estavam distribuídas radialmente em torno da origem do plano Cartesiano. Neste caso, apenas o algoritmo BURS apresentou um bom resultado. Quanto ao tempo de execução, nos dois exemplos o URS foi bem mais rápido que o BURS, em decorrência à dimensão do problema.

Também no caso bidimensional, aplicamos os métodos URS, BURS e CG, na reconstrução de imagens de dimensões 128×128 pixels, sendo estes, exemplos onde os pontos que queríamos calcular eram um subconjunto de pontos espaçados uniformemente. Os resultados de todos os métodos foram satisfatórios, porém, o tempo de execução dos algoritmos URS e CG foi muito maior do que o tempo de execução do algoritmo BURS, em todos os casos. No pior deles, o tempo do BURS não chegou a 15 minutos, enquanto os outros dois ultrapassaram 24 horas, sendo o URS ainda mais lento que o CG. Dessa maneira, concluímos que os métodos URS e CG, são computacionalmente impraticáveis em problemas como este. Todos os resultados numéricos aqui relacionados e comentados apresentam-se no Capítulo 3.

Referências Bibliográficas

Björck, Å. (1996). *Numerical Methods for Least Square Problems*. SIAM.

Briggs, W. L. and Henson, V. E. (1995). *The DFT: An Owner's Manual for the Discrete Fourier Transform*. SIAM.

Gearhart, W. B. and Shultz, H. S. (1990). The function $\frac{\sin x}{x}$. *The College Mathematics Journal*, 21:90–99.

Golub, G. and O'Leary, D. (1989). Some history of the conjugate gradient and Lanczos methods. *SIAM Review*, 31:50–102.

Golub, G. H. and Loan, C. F. V. (1996). *Matrix Computations*. Johns Hopkins University Press, Baltimore.

Klema, V. C. and Laub, A. J. (1980). The singular value decomposition: its computations and some applications. *IEEE Trans. Automat. Control*, AC-25:164–176.

Lawson, C. L. and Hanson, R. J. (1974). *Solving Least Square Problems*. Prentice–Hall.

Leite, L. W. B. (1998). *Introdução à Análise Espectral em Geofísica*. FADESP.

Oppenheim, A. V. and Schafer, R. W. (1989). *Discrete–Time Signal Processing*. Prentice–Hall.

Rosenfeld, D. (1998). An optimal and efficient new gridding algorithm using singular value decomposition. *Magnetic Resonance in Medicine*, 40:14–23.

Rosenfeld, D. (2002). New approach to gridding using regularization and estimation theory. *Magnetic Resonance in Medicine*, 48:193–202.

Tikhonov, A. N. and Arsenin, V. Y. (1977). *Solutions of Ill-posed Problems*. John Wiley & Sons.

Trefethen, L. N. and Bau, D. (1997). *Numerical Linear Algebra*. SIAM.

Apêndice A

Algoritmos

Apresentamos a seguir, os códigos em Matlab do método de reamostragem por blocos regularizada (rBURS), nos casos unidimensional e bidimensional. Lembramos que, para o algoritmo BURS, basta considerarmos o parâmetro de regularização igual a zero.

A.1 Caso Unidimensional

Com o objetivo de retomar a notação introduzida no Capítulo 2, consideremos uma amostra de uma função real contínua f em um conjunto finito de pontos $\{\tau_1, \tau_2, \ldots, \tau_m\}$, $\tau_i \in I\!R$, $i \in M = \{1, 2, \ldots, m\}$, não igualmente espaçados. O problema de reamostragem uniforme consiste em encontrar uma aproximação para a função em um conjunto de pontos uniformemente espaçados, i.e., aproximar $f(t_j)$, $t_j = t_0 + j\Delta t$, $t_0 \in I\!R$, $j \in N = \{1, 2, \ldots, n\}$.

Vimos anteriormente que este problema pode ser formulado na resolução de um sistema de equações lineares $Ax = b$ e que pode ser resolvido utilizando a pseudoinversa da matriz A. Também foi visto que, nos casos em que m e n são da ordem de milhares, o cálculo da pseudoinversa da matriz A é impraticável. A seguir, apresentamos o código em Matlab do algoritmo rBURS, que contorna este problema calculando por blocos uma matriz que aproxima a pseudoinversa de A.

```
function [ft,tempo] = rburs(r,tau,ftau,t);
% r = parâmetro de regularização.
% tau = vetor que contém pontos onde conhecemos a função.
% ftau = valores de f em tau (amostras).
% t = vetor que contém pontos igualmente espaçados onde queremos estimar
```

```
%     a função.
% OBS: Para o BURS, considerar r=0.
tic
tau = sort(tau);
t = sort (t);
dt = t(2) - t(1);
M = length(tau);
N = length(t);
%%%%%%%%%%%%%%%%%%%% MATRIZ A %%%%%%%%%%%%%%%%%%%%
B = t'*ones(1,M);
B = B';
C = tau'*ones(1,N);
A = (1/dt)*(C - B);
clear B;
clear C;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
delta = 5*dt;
DeltaT= 1.8*delta;
%%%%%%%%%% PSEUDO-INVERSA APROX. DE A %%%%%%%%%%
Ap = zeros(N, M);
for i = 1 :  N;
Jb = 0;
Ib = 0;
    for j = 1 :  M;
    d1 = norm( tau( j ) - t( i ) );
        if d1 < delta;
            h1 = length(Jb);
            Jb(h1 + 1) = j;
        end
    end
    for n = 1 :  N;
    d2 = norm( t(n) - t(i) );
        if d2 < DeltaT;
            h2 = length(Ib);
            Ib(h2 + 1) = n;
```

```
        end
        if n == i;
            p = h2;
        end
    end
h1 = length(Jb);
h2 = length(Ib);
Jb = Jb(2 :  h1);
Ib = Ib(2 :  h2);
Ab = A(min(Jb):max(Jb), min(Ib):max(Ib));    % Ab = submatriz de A
Ab = sinc(Ab);
Abp = pseudo(Ab,r,p);    % Abp = pseudoinversa de Ab
Ap( i , min(Jb):max(Jb)) = Abp;     % Ap = pseudoinversa aprox.  de A
clear p;
end
%%%%%% Solução %%%%%%
ft = Ap*ftau';
tempo = toc;
```

A.2 Caso Bidimensional

Retomemos também para este caso a notação introduzida no Capítulo 2. Consideremos uma amostra de uma função real contínua $f : I\!R^2 \to I\!R$, em um conjunto finito de pontos não igualmente espaçados, $\{(\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots, (\alpha_m, \beta_m)\}$. Nosso objetivo é encontrar uma aproximação para $f(a_i, b_j)$, $(a_i, b_j) = (a_0, b_0) + (i\Delta a, j\Delta b)$, $(a_0, b_0) \in I\!R^2$, $i \in N_1 = \{1, 2, \ldots, n_1\}$ e $j \in N_2 = \{1, 2, \ldots, n_2\}$.

Analogamente ao caso unidimensional, este problema pode ser formulado na resolução de um sistema de equações lineares $Ax = b$ e que pode ser resolvido mais uma vez utilizando a pseudoinversa da matriz A. Com o objetivo de facilitar a formulação do problema, ordenamos os pares ordenados (α_k, β_k), $k \in M = \{1, 2, \ldots, m\}$ em uma matriz τ, $m \times 2$ e os pontos Cartesianos unifomemente espaçados, por coluna, em uma matriz t, $n \times 2$, como segue: $t = (\ (\widehat{a}_1, \widehat{b}_1), (\widehat{a}_2, \widehat{b}_2), \ldots, (\widehat{a}_n, \widehat{b}_n)\ )$, de maneira que $(\widehat{a}_1, \widehat{b}_1) = (a_1, b_1)$, $(\widehat{a}_2, \widehat{b}_2) = (a_1, b_2)$, $(\widehat{a}_3, \widehat{b}_3) = (a_1, b_3), \ldots, (\widehat{a}_{n_2}, \widehat{b}_{n_2}) = (a_1, b_{n_2})$, $(\widehat{a}_{n_2+1}, \widehat{b}_{n_2+1}) = (a_2, b_1)$, $\ldots$, $(\widehat{a}_n, \widehat{b}_n) = (a_{n_1}, b_{n_2})$, onde $n = n_1 n_2$.

Assim, como no primeiro caso, quando m e n são da ordem de milhares, o cálculo da pseudoinversa da matriz A é impraticável. Apresentamos a seguir o código em Matlab do algoritmo anterior, adaptado ao caso bidimensional.

```
function [ft,tempo] = rblocos2d(r,tau,ftau,t,da,db,n1,n2)
% r = parâmetro de regularização.
% tau = matriz m×2 que contém as coordenadas dos pontos onde
%     conhecemos a funçao.
% ftau = valores de f em tau (amostras).
% t = matriz n×2 que contém as coordenadas dos pontos uniformemente
%     espaçados, onde queremos aproximar a função.
% da = espaçamento em x.
% db = espaçamento em y.
% n1 = número de elementos em x.
% n2 = número de elementos em y.
% OBS: Para o BURS, considerar r=0.
tic
M = length(tau);
N = length(t);
%%%%%%%%%%%%%%%%%%%% MATRIZ A %%%%%%%%%%%%%%%%%%%%
Bx = t(:,1)*ones(1,M);
Bx = Bx';
Cx = tau(:,1)*ones(1,N);
Ax = (1/da)*(Cx - Bx);
By = t(:,2)*ones(1,M);
By = By';
Cy = tau(:,2)*ones(1,N);
Ay = (1/db)*(Cy - By);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
delta = 3*da + 3*db;
DeltaT = 1.8*delta;
%%%%%%%%%%% PSEUDO-INVERSA APROX. DE A %%%%%%%%%%%
Ap = zeros(N,M);
for i = 1:N;
u = 1;
```

```
g = 1;
Ib = 0;
Jb = 0;
    for j = 1:M;
        d1 = norm(tau(j,:)  - t(i,:),inf);
        if d1 < delta
            Ib(u) = j;
            u = u + 1;
        end
    end
    for n = 1:N
        d2 = norm(t(n,:)  - t(i,:),inf);
        if d2 < DeltaT
            Jb(g) = n;
            g = g + 1;
        end
        if n==i
            p = g - 1;
        end
    end
mb = length(Ib);
nb = length(Jb);
    for n = 1 :  mb
        for j = 1 :  nb
            Abx(n,j) = Ax(Ib(n), Jb(j));
            Aby(n,j) = Ay(Ib(n), Jb(j));
        end
    end
Abx = sinc(Abx);
Aby = sinc(Aby);
Ab = Abx.*Aby;     % Ab = submatriz de A
Abp = pseudo(Ab,r,p);    % Abp = pseudoinversa de Ab
    for j = 1 :  mb
        Ap(i, Ib(j)) = Abp(j);    % Ap = pseudoinversa aprox.  de A
    end
```

```
clear p;
clear Abx;
clear Aby;
clear Abp;
clear Ab;
end
%%%%%% Solução %%%%%%
ft = Ap*b;
ft = reshape (ft,n2,n1);
tempo = toc;
```

Finalizamos este Apêndice com o código de um algoritmo, o qual chamamos de Pseudo, utilizado nos dois algoritmos vistos acima, que calcula a pseudoinversa regularizada das submatrizes de A.

```
function Abs=pseudo(A,r,p);
%r = parâmetro de regularização.
%p = linha de Abs que corresponde a ti.
[m,n] = size(A);
[U,S,V] = svd(A);
    if m > 1
        sigma = diag(S);
    else
        sigma = S(1);
    end
    for i = 1:length(sigma)
        d(i) = sigma(i)/(sigma(i)^2 + r);
    end
    if m>n
        D = diag(d);
        for j = n+1 :  m
            D(:,j)=zeros(n,1);
        end
        Abs = V(p,:)*D*U';
    else
```

```
        D = diag(d);
        for j = m+1:n;
            D(j,:)=zeros(1,m);
        end
        Abs = V(p,:)*D*U';
    end;
```