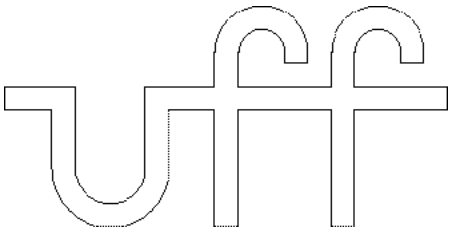
Apostila
de
Códigos de Programas Demonstrativos
para
Fundamentos de
Processamento Digital de Sinais

(Versão A2025M03D17)



Universidade Federal Fluminense

Alexandre Santos de la Vega

Departamento de Engenharia de Telecomunicações – TET

Escola de Engenharia – TCE

Universidade Federal Fluminense – UFF

Março – 2025

.

621.3192 (*) D278 (*) 2025	de la Vega, Alexandre Santos Apostila com Códigos de Programas Demonstrativos para Processamento Digital de Sinais / Alexandre Santos de la Vega. – Niterói: UFF/TCE/TET, 2025. 213 (sem romanos) ou 235 (com romanos) p. (*) Apostila com Códigos de Programas Demonstrativos – Graduação, Engenharia de Telecomunicações, UFF/TCE/TET, 2025. 1. Processamento de Sinais. 2. Processamento Digital de Sinais. 3. Telecomunicações. I. Título.

(*) OBTER INFO NA BIBLIOTECA, ATUALIZAR E PEDIR NOVO REGISTRO !!!

Aos meus alunos.

Prefácio

O trabalho em questão aborda os tópicos a serem apresentados na disciplina Fundamentos de Processamento Digital de Sinais. O material completo, dividido em apostilas e tutoriais, encontra-se dividido nos seguintes volumes:

1. O conteúdo teórico pode ser encontrado no volume entitulado Apostila de Teoria para Fundamentos de Processamento Digital de Sinais.

2. O conteúdo prático pode ser encontrado no volume entitulado Apostila de Códigos de Programas Demonstrativos para Fundamentos de Processamento Digital de Sinais.

3. As especificações dos trabalhos extra classe propostos na disciplina podem ser encontradas no volume entitulado Apostila de Trabalhos Extra Classe de Exercício e de Código (TEC) para Fundamentos de Processamento Digital de Sinais.

4. Conteúdos matemáticos básicos, necessários à disciplina em questão, são abordados no volume entitulado Apostila de Nivelamento para Fundamentos de Processamento Digital de Sinais.

5. Uma abordagem integradora dos tópicos de interesse da disciplina, de forma simples e direta, utilizando o sistema de média móvel como exemplo, pode ser encontrado no volume entitulado Tutorial sobre Sistema de Média Móvel para Fundamentos de Processamento Digital de Sinais.

6. Um conteúdo associado com Teoria de Sistemas, Teoria de Controle, Teoria de Circuitos e Teoria de Processamento de Sinais, envolvendo aspectos teóricos e códigos de programas demonstrativos, pode ser encontrado no volume entitulado Tutorial sobre Influência dos zeros e dos pólos da Função de Transferência no formato da função Resposta em Frequência para Fundamentos de Processamento Digital de Sinais.

Os documentos foram escritos com o intuito de servir como uma referência rápida para os alunos dos cursos de graduação e de mestrado em Engenharia de Telecomunicações da Universidade Federal Fluminense (UFF). O material básico utilizado para o conteúdo teórico foram as minhas notas de aula, que, por sua vez, originaram-se em uma coletânea de livros sobre os assuntos abordados. Por outro lado, os códigos de programas demonstrativos e as especificações dos trabalhos propostos são completamente autorais.

A motivação inicial para o desenvolvimento desse trabalho foi a de aumentar o dinamismo das aulas. Logo, deve ficar bem claro que os documentos produzidos não pretendem substituir os livros textos ou outros livros de referência. Pelo contrário, espera-se que eles sejam utilizados como ponto de partida para estudos mais aprofundados, utilizando-se a literatura existente.

Espero conseguir manter o presente texto em constante atualização e ampliação.

Correções e sugestões são sempre bem-vindas.

Alexandre Santos de la Vega
UFF / TCE / TET

Agradecimentos

Àqueles professores do Departamento de Engenharia de Telecomunicações (TET), da Escola de Engenharia (TCE), da Universidade Federal Fluminense (UFF), que colaboraram com críticas e sugestões bastante úteis à finalização da versão inicial deste trabalho.

Aos ex-funcionários do TET, Arlei, Carmen Lúcia, Eduardo Wallace, Francisco e Jussara, pelo apoio constante.

Aos meus alunos, que, além de servirem de motivação principal, obrigam-me sempre a me tentar melhorar, em todos os sentidos.

Mais uma vez, e sempre, aos meus pais, por tudo.

Alexandre Santos de la Vega
UFF / TCE / TET

Apresentação do material didático

Metodologia de construção

- O material aqui apresentado não é fruto de um projeto educacional envolvendo idealização, planejamento, pesquisa, estudo, estruturação, desenvolvimento, revisão e edição.
- Pelo contrário, ele nasceu, evoluiu e tem sido mantido de uma forma bem orgânica.

Histórico

- Em 1995, o autor ingressou no Departamento de Engenharia de Telecomunicações (TET) da Universidade Federal Fluminense (UFF) e, desde então, tem sido responsável por diversas disciplinas oferecidas pelo TET para o Curso de Engenharia de Telecomunicações, da Escola de Engenharia da UFF (TCE/UFF), e para o Curso de Ciência da Computação, do Instituto de Computação da UFF (IC/UFF).
- Na época do seu ingresso, o Processamento Digital de Sinais já era um assunto presente na área de Telecomunicações. E com importância crescente. Apesar disso, ainda não era oferecida pelo TET uma disciplina formal sobre a matemática que o fundamenta.
- Com essa percepção, ele criou a disciplina optativa "Introdução ao Processamento Digital de Sinais", em 1998.
- Para dar suporte às aulas, foram elaboradas as primeiras notas de aula (manuscritas) para a disciplina optativa criada no TET. Nessa primeira tentativa de implantação da disciplina, foi usada a referência [Mit98] como livro texto.
- A disciplina optativa foi oferecida pelo autor apenas durante dois períodos letivos, em virtude do seu afastamento para finalização do seu doutoramento.
- Durante o afastamento, e mesmo algum tempo depois, a disciplina optativa foi oferecida por outro professor do TET. Nesse período, o autor lançou uma outra disciplina optativa, vinculada à primeira, tratando do Projeto de Filtros Digitais.
- Tendo voltado a ministrar a disciplina, o autor decidiu ampliar as notas de aula manuscritas, baseando-se em diversos outros livros.
- Na primeira década de 2000, o TET realizou uma reforma curricular e a disciplina optativa "Introdução ao Processamento Digital de Sinais" tornou-se obrigatória, sob o nome de "Processamento Digital de Sinais".

- Em 2008, com os objetivos iniciais de melhor organizar os manuscritos e de atender aos apelos dos alunos por cópia dos manuscritos, eles foram apenas transcritos para o Sistema de Preparação de Documentos LaTeX [KD04] , [MG04]. Assim, surgiu a primeira versão da apostila de teoria.

- A partir daí, com a maturação gradual que a disciplina foi ganhando a cada período letivo, novos conteúdos foram surgindo. Ora por curiosidade do autor, procurando incorporar um determinado tópico na disciplina. Ora por curiosidade dos alunos, por demandarem algum assunto em especial. Ora por necessidade pedagógica, pois, ao se perceberem dúvidas recorrentes dos alunos, novas formas de abordagem têm sido testadas.

- Além disso, como filosofia educacional do autor, as questões que fazem parte de toda e qualquer forma de avaliação formal da disciplina (testes, provas, trabalhos) são anexadas ao conteúdo, na forma de exercícios propostos.

- No final da década de 2010, o TET realizou uma nova reforma curricular, a qual acarretou uma redução na quantidade e na carga horária das disciplinas. Isso provocou uma reformulação na abordagem dos tópicos da disciplina, que passou a ser denominada de "Fundamentos de Processamento Digital de Sinais".

- Ainda como filosofia educacional do autor, a apostila de teoria não apresenta figuras que ilustrem os assuntos abordados. Pelo contrário, é demandado aos alunos que eles gerem as suas próprias figuras, a partir de um aplicativo computacional adequado.

- Desde 2011, objetivando incentivar os alunos a modificarem códigos existentes e a gerarem seus próprios códigos, uma nova apostila tem sido elaborada, contendo códigos para programas demonstrativos, relativos aos tópicos abordados na apostila de teoria, em sala de aula e/ou em alguma forma de avaliação formal da disciplina.

- A partir de 2016, com a incorporação de trabalhos semanais na prática da disciplina, uma nova apostila tem sido elaborada, contendo os trabalhos extra classe (TEC) propostos a cada período letivo.

- Em 2018, foi percebido que, utilizando o sistema de média móvel como exemplo, é possível abordar e integrar os tópicos de interesse da disciplina de forma simples e direta. Além disso, com ele, também é possível gerar exemplos, exercícios e aplicações práticas, com certa facilidade. Assim, teve início a elaboração do tutorial sobre o sistema de média móvel.

- Em 2019, foi iniciado um tutorial sobre a influência dos zeros e dos pólos da Função de Transferência no formato da função Resposta em Frequência, buscando atender a uma série de motivações, listadas no documento em questão.

- A partir do isolamento social, imposto pela Pandemia de COVID-19, nos anos de 2020 e 2021, foi notada uma deficiência relativa aos conteúdos matemáticos básicos de Matrizes, Números complexos e Polinômios, necessários à disciplina em questão.

- A princípio, tais conteúdos foram apenas abordados em aulas iniciais de revisão. Em seguida, um texto inicial foi anexado à apostila de teoria, na forma de apêndices. Por fim, a partir de 2025, uma apostila de nivelamento, foi incorporada ao conjunto dos materiais autorais produzidos.

Comentários gerais:

- Conforme foi exposto acima, desde o início da sua confecção até o presente momento, sempre foram preparadas diversas versões de cada documento ao longo de um mesmo período letivo. Por essa razão, o identificador "Versão A<ano>M<mês>D<dia>" aparece logo abaixo do título de cada apostila.

- No tocante à apresentação do conteúdo teórico, os manuscritos originais continham apenas tópicos, destinados à abordagem do conteúdo programático durante as aulas. Pode-se dizer que tais manuscritos representavam apenas um roteiro de aula. Gradativamente, com a evolução da apostila de teoria, os tópicos têm sido trocados por textos dissertativos, relativos ao conteúdo abordado.

- No ponto de vista estrutural é que o aspecto dinâmico dos documentos mais se tem feito presente. Buscando uma melhor apresentação dos tópicos abordados, os diversos seccionamentos de texto (capítulos, seções, subseções, etc.) comumente surgem, são mesclados e desaparecem, a cada nova versão.

- Por tudo isso, pode-se asseguradamente dizer que todo o material produzido encontra-se em constante atualização.

- Na preparação das aulas, têm sido utilizados os seguintes livros:

  - Livros indicados pela ementa da disciplina: [DdSN10], [Mit98].
  - Outros livros indicados: [Rob09], [PM06], [Jac96], [She95], [SK89], [Ant86], [SDD84], [OWY83], [PL76], [OS75], [Cad73].

Teoria abordada no material didático

- Introdução <2 horas>
  - Conceitos básicos: que busca contextualizar a disciplina no âmbito do curso e apresentar conceitos que serão necessários ao longo do texto. <2 horas>
  - Conexão entre os modelos analógico e discreto/digital: que apresenta um resumo das representações dos sinais analógicos no domínio da freqüência e aborda as duas formas de conexão entre os domínios analógico e digital. [Opcional]
- Sinais e sistemas (com tempo discreto) no domínio do tempo <12 horas>
  - Sinais no domínio do tempo: definições, classificações, operações, exemplos e caracterizações. <4 horas>
  - Seqüências exponenciais: características relevantes de exponenciais, funções com dependência exponencial, decomposição de funções usando exponenciais, amostragem de sinais contínuos no tempo. <4 horas>
  - Sistemas no domínio do tempo: definições, classificações, operações, exemplos e caracterizações. <4 horas>
- Representações de um Sistema Linear e Invariante ao Tempo (SLIT) <10 horas>
  - Resposta ao impulso. <1 hora>
  - Diagramas de blocos de complexidade genérica. <1 hora>
  - Equação de diferença. <1 hora>
  - Diagramas de sistema (ou estruturas ou realizações). <2 horas>
  - Operador de transferência. <1 hora>
  - Diagrama de pólos e zeros do operador de transferência. <2 horas>
  - Equações de estado. <2 horas>
  - Relações e mapeamentos entre as diversas representações. <distribuído ao longo da apresentação do conteúdo e exercitado na forma de trabalhos>
- Respostas de um Sistema Linear e Invariante ao Tempo (SLIT) <10 horas>
  - Cálculos da resposta de um SLIT <8 horas>
    - Cálculo da resposta de um SLIT baseado na solução das equações de estado. <1 hora>
    - Cálculo da resposta de um SLIT baseado no uso do operador de transferência. <1 hora>

    - Cálculo da resposta de um SLIT baseado na solução convencional da equação de diferença. <4 horas>
    - Cálculo da resposta de um SLIT FIR (Resposta ao Impulso Finita) com entrada de comprimento indefinido. <2 horas>
  - Tipos de resposta de um SLIT <2 horas>
    - Resposta completa.
    - Resposta homogênea + resposta do sistema relaxado (resposta particular + resposta complementar).
    - Resposta ao estado + resposta à entrada.
    - Resposta natural + resposta forçada.
    - Resposta transitória + resposta permanente.

- Noções da representação em domínio transformado para sistemas de primeira ordem [Opcional]
  - Resposta em Freqüência: baseado no cálculo da resposta de um SLIT de primeira ordem, para um determinado tipo de sinal de entrada, pode-se identificar um novo tipo de representação para o sistema.
  - Função de Transferência: baseado no cálculo da resposta de um SLIT de primeira ordem, para um determinado tipo de sinal de entrada, pode-se identificar um novo tipo de representação para o sistema.

- Sinais e sistemas (com tempo discreto) no domínio da freqüência <20 horas>
  - Sinais <10 horas>
    - Motivações para a mudança de domínio de uma representação. <1/2 hora>
    - Revisão das representações em freqüência com tempo contínuo (Série de Fourier, Transformada de Fourier e Transformada de Laplace). <1/2 hora>
    - Série de Fourier de Tempo Discreto (DTFS). <1 hora>
    - Transformada de Fourier de Tempo Discreto (DTFT). <2 horas>
    - Transformada de Fourier Discreta (DFT). <2 horas>
    - Transformada Z. <2 horas>
    - Relações entre as diversas representações em freqüência, parâmetros e efeitos importantes. <2 horas>
  - Técnicas básicas para aceleração do cálculo da DFT. [Opcional]
  - Aplicações básicas da Transformada Z em sinais e sistemas. <6 horas>
    - Transformada Z de alguns sinais importantes.
    - Transformada Z aplicada na convolução.
    - Transformada Z aplicada em sinais deslocados.
    - Transformada Z aplicada na equação de diferença.
    - Transformada Z no cálculo de respostas de um SLIT.

  - SLIT de ordem qualquer <4 horas>
    * Tipos de respostas de um sistema.
    * Resposta completa em domínio transformado.
    * Resposta em Freqüência.
    * Seletividade em Freqüência.
    * Função de Transferência ou Função de Sistema.
    * Representações de um SLIT no domínio da freqüência.

- Aplicações: exemplos de aplicações são distribuídos ao longo do texto e exercitados na forma de trabalhos.

Objetivos da disciplina

- Apresentar a base matemática que fundamenta o Processamento Digital de Sinais.
- Trabalhar com sistemas que apresentem as seguintes características:
  - Sistema Linear e Invariante ao Tempo (SLIT).
  - Sistema *Single-Input Single-Output* (SISO).
  - Sistema operando com tempo discreto.
  - Sistema operando com sinais definidos em tempo discreto, quantizados (digitais) ou não (amostrados).
- Trabalhar com sinais básicos que sejam simultaneamente dependentes das variáveis tempo e freqüência, utilizando-os na composição dos demais sinais envolvidos.
- Discutir a análise de sistemas no domínio da variável tempo e no domínio da variável freqüência. No domínio do tempo, o foco está na FORMA que os sinais apresentam. No domínio da freqüência, o foco está na COMPOSIÇÃO que os sinais apresentam.
- Discutir a aplicação dos conceitos de Operador de Transferência (no domínio do tempo) e de Função de Transferência (no domínio da freqüência), bem como a relação existente entre ambos.
- Discutir a aplicação do conceito de estado de um sistema e da análise do sistema no espaço de estados.

Sumário

Parte I

Introdução

Capítulo 1

Ambiente de simulação matemática

1.1 Introdução

Neste capítulo, são apresentados alguns conceitos básicos relativos ao ambiente de simulação matemática utilizado. São abordados os seguintes itens: aspectos gerais sobre o ambiente (definições; interação com usuário; interpretador de comandos; *namespace* e *workspace*; tipos de dados; estruturas de dados; comandos e funções comumente utilizados; ações típicas de criação/acesso/operações relativas a estruturas de dados), representação gráfica bidimensional (curvas), representação gráfica tridimensional (curvas e superfícies), representação gráfica tridimensional (imagens), simulação de equação de diferença.

1.2 Aspectos gerais sobre o ambiente

1.2.1 Definições

- O Octave é um aplicativo computacional.
- O Octave pode ser interpretado como uma máquina computacional virtual, que entende uma linguagem própria e a traduz para uma outra linguagem, que é entendida por uma outra máquina computacional, hierarquicamente abaixo dela.
- O Octave trabalha com uma linguagem do tipo imperativa. A tradução realizada é uma interpretação. Assim, cada comando da linguagem representa uma assertiva específica, que deve ser dinamicamente traduzida e executada.
- Um conjunto composto por comandos que sejam reconhecidos pelo aplicativo é denominado de programa ou código.
- Como nas demais linguagens de programação, os comandos são formados a partir de um conjunto de caracteres, formado por:
  - Um alfabeto natural: {A,B,C, ..., X, Y, Z} e {a,b,c, ..., x, y, z}.
  - Dígitos decimais: {0, 1, ..., 8, 9}.
  - Caracteres especiais, tais como:

    `% ' " $ # ; , . : + - * / \ ^ ! & | ? < > ( ) [ ] { }`

- A linguagem que é reconhecida pelo Octave é do tipo *case sensitive*, o que significa que identificadores similares, porém com letras minúsculas ou maiúsculas, são considerados identificadores diferentes.

- As seguintes palavras possuem significado especial para a linguagem reconhecida pelo Octave e, por isso, são ditas palavras reservadas (ou *reserved words* ou *keywords*) pela linguagem: break, case, catch, continue, else, elseif, end, for, function, global, if, otherwise, persistent, return, switch, try, while.

1.2.2 Interação com usuário

- A parte do Octave que interage com o usuário é denominada de interpretador de comandos (ou *shell*). Os comandos podem ser passados para o interpretador de comandos de duas formas básicas: interativamente ou por lote (*batch*).

- Na forma interativa, os comandos são passados individualmente para o interpretador de comandos, por digitação direta.

- Na forma por lote, os comandos são organizados em um arquivo e o nome do arquivo é passado ao interpretador de comandos, que se encarrega de capturar cada um dos comandos do arquivo. Esse arquivo, contendo um lote de comandos, recebe a denominação de *script*. De uma forma geral, os comandos são organizados em um arquivo do tipo TEXTO, denominado de *M-file* e identificado como "nome.m".

- Um conjunto de comandos que é repetidamente utilizado por um programa e/ou é utilizado por diversos programas pode ser organizado em uma função. Do ponto de vista do usuário, uma função é, basicamente, um *script* com uma sintaxe específica.

- As estruturas de dados em Octave são dinamicamente tipadas. Não se faz necessária uma declaração antecipada, para a definição de tipo da estrutura e para a reserva de espaço em memória. Basta um simples comando de construção (construtor) ou uma simples atribuição de valores, em tempo de execução, para criá-las ou modificá-las.

1.2.3 Interpretador de comandos

- O interpretador de comandos (ou *shell*) é a parte do aplicativo que interage com o usuário.

- Alguns comandos ecoam seu resultado para a janela do interpretador de comandos. Para evitar esse eco, basta adicionar o caractere ';' ao final do comando.

- Caso uma expressão seja avaliada sem uma atribuição explícita, a variável padrão 'ans' (*answer* ou resposta) recebe o resultado da avaliação.

- Comentários (texto não traduzido) podem ser gerados com o auxílio do caractere '%', dado que, a partir dele, tudo é ignorado até o final da linha de comando.

- Caso um comando seja muito longo, ele pode ser quebrado em várias linhas, adicionando-se reticências (três pontos sem espaço entre eles) ao final de cada linha. Cabe ressaltar que um comentário de linha não pode ser quebrado e continuado.

- Uma linha de código pode conter vários comandos, que devem ser separados por um dos seguintes caracteres: ',' e ';'. Como citado anteriormente, resultados de comandos terminados com ';' não são ecoados. Podem ser inseridos espaços antes e depois dos caracteres de separação.

- Comandos úteis na interação com o interpretador de comandos:
iskeyword, lookfor 'padrão', help 'nome', who, whos, clear, close, clc.

- Funções úteis na interação com o interpretador de comandos: disp(), pause().

- Comandos úteis para identificar funções do tipos *built-in* ou *M-file*:
which 'função', exist 'função' ('built-in' = 5), builtin('function', arg1, arg2, ..., argN).

- Funções relacionadas com a versão do computador e o computador: version, computer.

1.2.4 *Namespace* e *workspace*

- Em ambientes computacionais, um *namespace* é um mecanismo que envolve um conjunto de sinais, símbolos ou nomes, a fim de identificar diferentes objetos, de maneira única, simples e direta.

- Comumente, os *namespaces* são organizados de foram hierárquica, possibilitando o reuso de nomes.

- O Octave possui o seu próprio *namespace*, onde são definidos variáveis e valores padrões. Por exemplo: a unidade imaginária (i ou j) e os números 'e=2.7183...' e 'pi=3.1416...'.

- Ao iniciar uma sessão de trabalho (ou ambiente ou *environment*), o interpretador de comandos cria o seu próprio *namespace*, que é denominado de *workspace* e que se encontra hierarquicamente abaixo do *namespace* geral do Octave.

- Dessa forma, é possível definir novas variáveis e novos valores, diferentes daqueles definidos pelo Octave, utilizando os mesmos identificadores, mas sem destruir os objetos originais.

- Porém, deve ficar claro que, ao se criar novos objetos no ambiente de trabalho, com os mesmos identificadores originais, apenas os novos objetos serão acessados. Para voltar a acessar os objetos padrões, devem-se destruir os objetos definidos no ambiente (utilizando o comando 'clear').

1.2.5 Tipos de dados

- O Octave admite os seguintes tipos de dados:
  - Logical: valores lógicos FALSE (igual a zero) e TRUE (diferente de zero).
  - Char: valores associados com caracteres textuais.
  - Numérico inteiro: valores interpretados sem sinal (uint8, uint16, uint32, uint64) e valores interpretados com sinal (int8, int16, int32, int64).
  - Numérico real (*float*): single e double.

- As quantidades numéricas são representadas por um Sistema de Numeração Posicional Convencional (SNPC), com base $b = 2$, codificado em ponto flutuante (*floating point*).

- Em cálculos numéricos, independentemente dos tipos de dados dos operandos, os cálculos são sempre efetuados na maior representação numérica, que é double. Isso é denominado de aritmética de precisão dupla.
- Variáveis predefinidas, que contêm valores especiais, associados com tipos numéricos: eps, bitmax, realmax, realmin, intmax, intmin, inf, NaN ou nan, i ou j, pi.
- Funções relacionadas a números complexos:
  complex(), conj(), real(), imag(), abs(), angle(), unwrap().
- Função para controle da exibição de dado numérico: format().

1.2.6 Estruturas de dados

- A estrutura de dados básica no Octave é MATRIZ, que é uma estrutura bidimensional retangular.
- Todos os elementos em uma MATRIZ são do mesmo tipo.
- Outras estruturas são definidas a partir de MATRIZ.
- As estruturas mais simples são: escalar (matriz 1x1), vetor linha (matriz 1xC), vetor coluna (matriz Lx1), matriz retangular (matriz LxC), matriz quadrada (matriz MxM), matriz multidimensional (matriz D1xD2xD3x...xDM).
- No caso de uma matriz tridimensional (matriz D1xD2xD3), as dimensões D1, D2 e D3 são respectivamente denominadas de linha, coluna e página.
- Uma dimensão Dn de valor unitário (Dn = 1) é denominada de SINGLETON.
- As estruturas mais complexas são arranjos multidimensionais denominados genericamente de ARRAY.
- Dois tipos de ARRAY são definidos:
  - *Cell array*: arranjo multidimensional de matrizes não necessariamente iguais.
  - *Structure array*: arranjo multidimensional de uma estrutura de dados similar a STRUCTURE em C e a RECORD em Pascal, onde diferentes campos são definidos por um nome e por uma estrutura de dados.

1.2.7 Funções básicas dedicadas a matrizes

- Funções relacionadas com tipos especiais de matrizes:
  ones(), zeros(), eye(), diag(), magic(), rand(), randn().
- Funções relacionadas com a concatenação de matrizes:
  cat(), horzcat(), vertcat(), repmat(), blkdiag().
- Funções relacionadas com mudanças na forma de uma matriz:
  reshape(), rot90(), fliplr(), flipud(), flipdim(), transpose(), ctranspose(), permute().
- Funções relacionadas com deslocamento e ordenação de dados em uma matriz:
  circshift(), sort().
- Funções relacionadas com informações sobre uma matriz:
  ndims(), numel(), size(), length(), find().

1.2.8 Códigos

- O Código 1.1 apresenta diversos exemplos de criação de estruturas de dados.
- O Código 1.2 apresenta diversos exemplos de acesso a estruturas de dados.
- O Código 1.3 apresenta diversos exemplos de operações com estruturas de dados.

Código 1.1: Exemplos de criação de estruturas de dados.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: Aula_1_Lab_Octave_Criacao.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
% Titulo:                                                                       %
% Exemplos de criacao de estruturas de dados                                    %
%                                                                               %
%                                                                               %
% Autor : Alexandre Santos de la Vega                                           %
% Datas : /2022-01/                                                             %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% limpeza do ambiente de trabalho
%
clear all
close all


disp(' ')
disp(' ')
disp('          ==========================================')
disp('          Exemplos de criação de estruturas de dados')
disp('          ==========================================')
disp(' ')


%
% Exemplos de escalar (matriz 1x1)
%

disp(' ')
disp(' ')
disp('     ================================')
disp('     Exemplos de escalar (matriz 1x1)')
disp('     ================================')
disp(' ')

v = 'a'

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()
```

```
51 v = 4

53 disp(' ')
   disp('Pressione qualquer tecla para continuar...')
55 pause()

57 v = -8.5

59 disp(' ')
   disp('Pressione qualquer tecla para continuar...')
61 pause()

63 v = 3 + i*7

65 disp(' ')
   disp('Pressione qualquer tecla para continuar...')
67 pause()

69
   %
71 % Exemplos de vetor linha (matriz 1xC)
   %
73
   disp(' ')
75 disp(' ')
   disp('    ========================================')
77 disp('     Exemplos de vetor linha (matriz 1xC)')
   disp('    ========================================')
79 disp(' ')

81 v = 'abc'

83 disp(' ')
   disp('Pressione qualquer tecla para continuar...')
85 pause()

87 v = [4 12 100]

89 disp(' ')
   disp('Pressione qualquer tecla para continuar...')
91 pause()

93 v = [-8.5, 4, 11.3]

95 disp(' ')
   disp('Pressione qualquer tecla para continuar...')
97 pause()

99 v = [3 + i*7, -2 + i*9, i*11]

101 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
103 pause()

105 v = 5:11

107 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
```

```
109 pause()

111 v = 12:3:22

113 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
115 pause()

117 v = 78:-5:49

119 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
121 pause()

123
    %
125 % Exemplos de vetor coluna (matriz Lx1)
    %
127
    disp(' ')
129 disp(' ')
    disp('    ========================================')
131 disp('     Exemplos de vetor coluna (matriz Lx1)')
    disp('    ========================================')
133 disp(' ')

135 v = ['a';'b';'c']

137 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
139 pause()

141 v = [4 ; 12 ; 100]

143 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
145 pause()

147 v = [-8.5 ; 4 ; 11.3]

149 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
151 pause()

153 v = [3 + i*7 ; -2 + i*9 ; i*11]

155 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
157 pause()

159 v = [5:11]'

161 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
163 pause()

165 v = [12:3:22]'

167 disp(' ')
```

```
    disp('Pressione qualquer tecla para continuar...')
169 pause()

171 v = [76:-5:49]'

173 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
175 pause()

177
    %
179 % Exemplos de matriz retangular (matriz LxC)
    %
181
    disp(' ')
183 disp(' ')
    disp('     ==========================================')
185 disp('     Exemplos de matriz retangular (matriz LxC)')
    disp('     ==========================================')
187 disp(' ')

189 v = ['abcd'; 'efgh']

191 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
193 pause()

195 v = ['ab'; 'cd' ; 'ef' ; 'gh']

197 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
199 pause()

201 v = [4 12 100 1 ; 7 11 15 3]

203 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
205 pause()

207 v = [4 12 ; 100 1 ; 7 11 ; 15 3]

209 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
211 pause()

213 v = [1:6 ; 11:16 ; 21:26]

215 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
217 pause()

219
    %
221 % Exemplos de matriz quadrada M (matriz MxM)
    %
223
    disp(' ')
225 disp(' ')
    disp('     ==========================================')
```

```
227 disp('      Exemplos de matriz quadrada M (matriz MxM)')
    disp('      ==========================================')
229 disp(' ')

231 v = ['abcd'; 'efgh' ; 'ijkl' ; 'mnop']

233 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
235 pause()

237 v = [4 12 100 1 ; 7 11 15 3 ; 2 6 19 33 ; 8 10 44 56]

239 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
241 pause()

243 v = [1:4 ; 11:14 ; 21:24 ; 31:34]

245 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
247 pause()

249
    %
251 % Exemplos de cell array
    %
253
    disp(' ')
255 disp(' ')
    disp('      ======================')
257 disp('      Exemplos de cell array')
    disp('      ======================')
259 disp(' ')

261 %
    % Construção com atribuição
263 %

265 disp(' ')
    disp(' ')
267 disp('  ============================')
    disp('  Construção com atribuição')
269 disp('  ============================')
    disp(' ')
271
    v = {       1 ,                    [2 , 3] ;
273       [4 ; 5] , [6 , 7 ; 8 , 9 ; 10 , 11]
      }
275
    disp(' ')
277 disp('Pressione qualquer tecla para continuar...')
    pause()
279
    v{1,1}
281
    disp(' ')
283 disp('Pressione qualquer tecla para continuar...')
    pause()
285
```

```
v{1,2}

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

v{2,1}

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

v{2,2}

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

%
% Construção com função
%

disp(' ')
disp(' ')
disp('  ====================')
disp('  Construção com função')
disp('  ====================')
disp(' ')

v = cell(3,2)

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

v{1,1} = 1.1;
v{1,2} = 'a';
v{2,1} = [2 + i*3 , 4 + i*5];
v{2,2} = [6 + i*7 ; 8 + i*9];
v{3,1} = ['bcd' ; 'efg'];
v{3,2} = [10 , 11 ; 12 , 13 ; 14 , 15]

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

v{1,1}

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

v{1,2}

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

v{2,1}
```

```
345
    disp(' ')
347 disp('Pressione qualquer tecla para continuar...')
    pause()
349
    v{2,2}
351
    disp(' ')
353 disp('Pressione qualquer tecla para continuar...')
    pause()
355
    v{3,1}
357
    disp(' ')
359 disp('Pressione qualquer tecla para continuar...')
    pause()
361
    v{3,2}
363
    disp(' ')
365 disp('Pressione qualquer tecla para continuar...')
    pause()
367

369 %
    % Exemplos de structure array
371 %

373 disp(' ')
    disp(' ')
375 disp('     ============================')
    disp('      Exemplos de structure array')
377 disp('     ============================')
    disp(' ')
379
    %
381 % Construção com atribuição
    %
383
    disp(' ')
385 disp(' ')
    disp('  ========================')
387 disp('  Construção com atribuição')
    disp('  ========================')
389 disp(' ')

391 %
    clear v
393
    v.nome     = 'Aluno 1';
395 v.notas    = [1.1 2.1 3.1];
    v.situacao = 'Reprovado'
397
    disp(' ')
399 disp('Pressione qualquer tecla para continuar...')
    pause()
401
    v(2).nome     = 'Aluno 11';
403 v(2).notas    = [4.1 5.1 6.1];
```

```
v(2).situacao = 'VS'

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

v(3).nome     = 'Aluno 111';
v(3).notas    = [7.1 8.1 9.1];
v(3).situacao = 'Aprovado'

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

v(1)

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

v(2)

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

v(3)

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

%
% Construção com função
%

disp(' ')
disp(' ')
disp('  ====================')
disp('   Construção com função')
disp('  ====================')
disp(' ')

v = struct(      'nome' , { 'Aluno Nota Baixa'        , ...
                            'Aluno Nota Mediana'     , ...
                            'Aluno Nota Suficientemente alta' ...
                          } , ...
                'media' , { 3.9 , 5.9 , 6.1 } , ...
              'situacao' , { 'Reprovado', 'VS', 'Aprovado' } ...
           )

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

v(1)

disp(' ')
disp('Pressione qualquer tecla para continuar...')
```

```
463 pause()

465 v(2)

467 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
469 pause()

471 v(3)

473 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
475 pause()

477 %
    % EOF
479 %
```

Código 1.2: Exemplos de acesso a estruturas de dados.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %
3  % Arquivo: Aula_1_Lab_Octave_Acesso.m
   %
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

7
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9  %                                                                             %
   % Titulo:                                                                     %
11 % Exemplos de acesso a estruturas de dados                                    %
   %                                                                             %
13 %                                                                             %
   % Autor : Alexandre Santos de la Vega                                        %
15 % Datas : /2022-01/                                                          %
   %                                                                             %
17 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

19
   % limpeza do ambiente de trabalho
21 %
   clear all
23 close all

25
   disp(' ')
27 disp(' ')
   disp('          ==========================================')
29 disp('           Exemplos de acesso a estruturas de dados')
   disp('          ==========================================')
31 disp(' ')

33
   %
35 % Exemplos de vetor linha (matriz 1xC)
   %
37
```

```
disp(' ')
disp(' ')
disp('     ======================================')
disp('      Exemplos de vetor linha (matriz 1xC)')
disp('     ======================================')
disp(' ')

vl = 1:2:13

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

vl(3)

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

vl([2 , 5])

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

vl(4:6)

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()


%
% Exemplos de vetor coluna (matriz Lx1):
%

disp(' ')
disp(' ')
disp('     ======================================')
disp('      Exemplos de vetor coluna (matriz Lx1)')
disp('     ======================================')
disp(' ')

vc = [2:2:16]'

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

vc(6)

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

vc([1 , 4])

disp(' ')
disp('Pressione qualquer tecla para continuar...')
```

```
 97 pause()

 99 vc(3:5)

101 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
103 pause()

105
    %
107 % Exemplos de matriz retangular (matriz LxC)
    %
109
    disp(' ')
111 disp(' ')
    disp('     ==========================================')
113 disp('      Exemplos de matriz retangular (matriz LxC)')
    disp('     ==========================================')
115 disp(' ')

117 mlc = randn(6,4)

119 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
121 pause()

123 mlc(5,3)

125 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
127 pause()

129 %  ( L*(c-1) ) + l  =
    %  ( 6*(3-1) ) + 5  =  12 + 5  =  17
131 %
    mlc(17)
133
    disp(' ')
135 disp('Pressione qualquer tecla para continuar...')
    pause()
137
    mlc(2 , 1:3)
139
    disp(' ')
141 disp('Pressione qualquer tecla para continuar...')
    pause()
143
    mlc(2:5 , 3)
145
    disp(' ')
147 disp('Pressione qualquer tecla para continuar...')
    pause()
149
    mlc(3 , :)
151
    disp(' ')
153 disp('Pressione qualquer tecla para continuar...')
    pause()
155
```

```
    mlc(:  ,  2)
157
    disp('  ')
159 disp('Pressione qualquer tecla para continuar...')
    pause()
161
    mlc([1  3  5]  ,  [2  4])
163
    disp('  ')
165 disp('Pressione qualquer tecla para continuar...')
    pause()
167
    mlc(2:4  ,  2:3)
169
    disp('  ')
171 disp('Pressione qualquer tecla para continuar...')
    pause()
173

175 %
    % Exemplos de matriz tridimensional (matriz LxCxP)
177 %

179 disp('  ')
    disp('  ')
181 disp('     ==========================================================')
    disp('      Exemplos de matriz tridimensional (matriz LxCxP)')
183 disp('     ==========================================================')
    disp('  ')
185
    mlcp = randn(6,4,3)
187
    disp('  ')
189 disp('Pressione qualquer tecla para continuar...')
    pause()
191
    mlcp(5,3,2)
193
    disp('  ')
195 disp('Pressione qualquer tecla para continuar...')
    pause()
197
    %  ( L*C*(p-1) ) + ( L*(c-1) ) + l  =
199 %  ( 6*4*(2-1) ) + ( 6*(3-1) ) + 5  =  24 + 12 + 5  =  41
    %
201 mlcp(41)

203 disp('  ')
    disp('Pressione qualquer tecla para continuar...')
205 pause()

207 mlcp(2  ,  1:3  ,  1)

209 disp('  ')
    disp('Pressione qualquer tecla para continuar...')
211 pause()

213 mlcp(2:5  ,  3  ,  2)
```

```
215 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
217 pause()

219 mlcp([1 3 5] , [2 4] , 3)

221 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
223 pause()

225 mlcp(2:4 , 2:3 , :)

227 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
229 pause()

231
    %
233 % Exemplos de cell array
    %
235
    disp(' ')
237 disp(' ')
    disp('    ======================')
239 disp('     Exemplos de cell array')
    disp('    ======================')
241 disp(' ')

243 %
    % Construção com atribuição
245 %

247 disp(' ')
    disp(' ')
249 disp(' ========================')
    disp(' Construção com atribuição')
251 disp(' ========================')
    disp(' ')
253
    c = {       1 ,                    [2 , 3] ;
255       [4 ; 5] , [6 , 7 ; 8 , 9 ; 10 , 11]
      }
257
    disp(' ')
259 disp('Pressione qualquer tecla para continuar...')
    pause()
261
    c{1,1}
263
    disp(' ')
265 disp('Pressione qualquer tecla para continuar...')
    pause()
267
    c{1,2}
269
    disp(' ')
271 disp('Pressione qualquer tecla para continuar...')
    pause()
273
```

```
c{2,1}

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

c{2,2}

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

c{1,2}(1,1)

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

c{2,1}(2,1)

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

c{2,2}(2,:)

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()


%
% Exemplos de structure array
%

disp(' ')
disp(' ')
disp('    ============================')
disp('     Exemplos de structure array')
disp('    ============================')
disp(' ')

%
% Construção com atribuição
%

disp(' ')
disp(' ')
disp(' ==========================')
disp('  Construção com atribuição')
disp(' ==========================')
disp(' ')


s.nome     = 'Aluno 1';
s.notas    = [1.1 2.1 3.1];
s.situacao = 'Reprovado'

disp(' ')
```

```
333 disp('Pressione qualquer tecla para continuar...')
    pause()
335
    s(2).nome     = 'Aluno 11';
337 s(2).notas    = [4.1 5.1 6.1];
    s(2).situacao = 'VS'
339
    disp(' ')
341 disp('Pressione qualquer tecla para continuar...')
    pause()
343
    s(3).nome     = 'Aluno 111';
345 s(3).notas    = [7.1 8.1 9.1];
    s(3).situacao = 'Aprovado'
347
    disp(' ')
349 disp('Pressione qualquer tecla para continuar...')
    pause()
351
    s(1)
353
    disp(' ')
355 disp('Pressione qualquer tecla para continuar...')
    pause()
357
    s(2)
359
    disp(' ')
361 disp('Pressione qualquer tecla para continuar...')
    pause()
363
    s(3)
365
    disp(' ')
367 disp('Pressione qualquer tecla para continuar...')
    pause()
369
    s(1).nome
371
    disp(' ')
373 disp('Pressione qualquer tecla para continuar...')
    pause()
375
    s(2).notas(2)
377
    disp(' ')
379 disp('Pressione qualquer tecla para continuar...')
    pause()
381
    s(3).situacao
383
    disp(' ')
385 disp('Pressione qualquer tecla para continuar...')
    pause()
387
    %
389 % EOF
    %
```

Código 1.3: Exemplos de operações a estruturas de dados.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: Aula_2_Lab_Octave_Operacoes.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                       %
% Titulo:                                               %
% Exemplos de operações com estruturas de dados %
%                                                       %
%                                                       %
% Autor : Alexandre Santos de la Vega                   %
% Datas : /2022-01/                                     %
%                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% limpeza do ambiente de trabalho
%
clear all
close all


disp(' ')
disp(' ')
disp('            ==============================================')
disp('            Exemplos de operações com estruturas de dados')
disp('            ==============================================')
disp(' ')


%
% Exemplos de escalar (matriz 1x1)
%

e = ( -4 *          (3^2) ) + ( 6 *      (125^(1/3)) )

e = ( -4 * power(3,2) ) + ( 6 * nthroot(125,3) )


%
% Exemplos de vetor linha (matriz 1xC)
%

vl = [ 1 2 3 ]

emvl = 4 * vl

vldde = vl / 2

vldee = 2 \ vl

vlmvl = vl .* vl

% Erro !!!
```

```
%
% vlmvl = vl * vl

vlddvl = vl ./ vl

vldevl = vl .\ vl

vlee = vl .^ 2

vlevl = vl .^ vl


%
% Exemplos de vetor coluna (matriz Lx1)
%

vc = [ 1 2 3 ].'

emvc = 4 * vc

vcdde = vc / 2

vcdee = 2 \ vc

vcmvc = vc .* vc

% Erro !!!
%
% vcmvc = vc * vc

vcddvc = vc ./ vc

vcdevc = vc .\ vc

vcee = vc .^ 2

vcevc = vc .^ vc


%
% Exemplos de vetor linha e vetor coluna
%

vl = [ 1 3 5 ]

vc = [ 6 ; 4 ; 2]

vlpevc = vl * vc

% Erro !!!
%
% vlmvc = vl .* vc

vlx = [ 1+j   3+j*3 5+j*5 ]

vlxtc = vlx'

vlxt  = vlx.'
```

```
    vcx = [ 2+j*2 ; 4+j*4 ; 6+j*6 ]
118
    vcxtc = vcx'
120
    vcxt  = vcx.'
122


124
    %
126 % Exemplos de vetor linha/coluna e matriz retangular (matriz LxC)
    %
128
    vl = [ 1 3 ]
130
    vc = [ 6 ; 4 ; 2]
132
    mr = [ 10 20 30 ; 40 50 60 ]
134
    vlmmr = vl * mr
136
    mrmvc = mr * vc
138

140 %
    % Exemplos de matriz retangular (matriz LxC)
142 %

144 mr = [ 1 2 3 ; 4 5 6 ]

146 mrmmr = mr .* mr

148 mree = mr .^ 3

150 mrmmrt = mr    * mr.'

152 mrtmmr = mr.' * mr

154 mrx = [   1   2+j*2 ;   j*3 -4+j*4 ;
           -5 -6-j*6 ; -j*7   8-j*8 ]
156
    mrxj = conj(mrx)
158
    real(mrx)
160
    imag(mrx)
162
    abs(mrx)
164
    angle(mrx)
166
    angle(mrx)*180/pi
168
    %
170 % EOF
    %
```

1.3 Representação gráfica bidimensional (curvas)

1.3.1 Representação gráfica discreta de função unidimensional

- Um gráfico discreto de função unidimensional $y(x) = f(x)$ pode ser visto como uma sucessão de pontos $p = \{p_k\}$, formados por duplas $p_k = (x_k, y_k)$.
- Assim, o processo de se elaborar o gráfico pode ser definido como:
  - Montar os vetores $x = \{x_k\}$ e $y = \{y_k\}$.
  - Montar duplas $p_k = (x_k, y_k)$.
  - Definir um padrão representativo para um ponto em um espaço bidimensional.
  - Desenhar os pontos p_k no espaço bidimensional.

1.3.2 Representação gráfica contínua de função unidimensional

- Em um ambiente digital, não é possível traçar um gráfico contínuo.
- Porém, a partir de pontos do gráfico contínuo, pode-se obter uma aproximação interpolada.

1.3.3 Funções comumente utilizadas

- Funções relacionadas com gráficos:
  figure(), subplot(), axis(), hold on/off, title(), xlabel(), ylabel(), zlabel(), linspace(), logspace().
- Funções relacionadas com gráficos bidimensionais genéricos:
  help graph2d, stem(), stairs(), plot().
- Funções relacionadas com gráficos bidimensionais especializados:
  help specgraph, bar(), barh(), area(), pie(), hist(), rose(), contour(), contourf(), clabel(), compass(), feather(), quiver().

1.3.4 Códigos

- O Código 1.4 apresenta um exemplo de gráfico discreto de função unidimensional.
- O Código 1.5 apresenta um exemplo de criação de subjanelas em uma mesma tela.
- O Código 1.6 apresenta um exemplo de superposição de gráficos discretos.
- O Código 1.7 apresenta um exemplo de criação de subjanela ocupando vários espaços.
- O Código 1.8 apresenta um exemplo de criação de curva discreta, de curva contínua por meio de interpolador de ordem 0 e de curva contínua por meio de interpolador de ordem 1.

Código 1.4: Exemplo de gráfico discreto de função unidimensional.

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %
3 % Arquivo: Aula_3_Lab_Octave_Curvas_1.m
  %
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                        %
% Titulo:                                                %
% Exemplo de gráfico discreto de função unidimensional   %
%                                                        %
%                                                        %
% Autor : Alexandre Santos de la Vega                    %
% Datas : /2022-01/                                      %
%                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% limpeza do ambiente de trabalho
clear all
close all

% definicao da abscissa
n  = -5:5;

% definicao da ordenada
a2 = 1;
a1 = 0;
a0 = 0;
y  = a2.*(n.^2) + a1.*n + a0;

% criacao de uma tela de desenho (canvas)
figure (1)

% criacao de desenho
stem(n,y)

% criacao de labels
xlabel('n')
ylabel('y[n] = a_2 n^2 + a_1 n + a_0')

% criacao de titulo
title('Gráfico discreto de função unidimensional')

% controle de faixas
faixa = [-10 10 -5 30];
axis(faixa)

%
%EOF
%
```

Código 1.5: Exemplo de criação de subjanelas em uma mesma tela.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: Aula_3_Lab_Octave_Curvas_2.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
% Titulo:                                                                       %
% Exemplo de criação de subjanelas em uma mesma tela                            %
%                                                                               %
%                                                                               %
% Autor : Alexandre Santos de la Vega                                           %
% Datas : /2022-01/                                                             %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% limpeza do ambiente de trabalho
clear all
close all

% definicao da abscissa
n = -1.5:0.01:1.5;

% definicao da ordenada
y1 = (n);
y2 = (n.^2);
y3 = (n.^3);
y4 = (n.^4);

% criacao de uma tela de desenho (canvas)
figure (2)

% criacao de subjanela na tela
subplot(2,2,1)

% criacao de desenho
stem(n,y1,'b')

% criacao de titulo
title('Polinômios de grau ímpar')

% criacao de labels
ylabel('y[n] = n^1')

% criacao de subjanela na tela
subplot(2,2,2)

% criacao de desenho
stem(n,y2,'g')

% criacao de titulo
title('Polinômios de grau ímpar')

% criacao de labels
```

```
ylabel('y[n] = n^2')

% criacao de subjanela na tela
subplot(2,2,3)

% criacao de desenho
stem(n,y3,'r')

% criacao de labels
ylabel('y[n] = n^3')
xlabel('n')

% criacao de subjanela na tela
subplot(2,2,4)

% criacao de desenho
stem(n,y4,'k')

% criacao de labels
ylabel('y[n] = n^4')
xlabel('n')


%
%EOF
%
```

Código 1.6: Exemplo de superposição de gráficos discretos.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: Aula_3_Lab_Octave_Curvas_3.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                         %
% Titulo:                                                 %
% Exemplo de superposição de gráficos discretos           %
%                                                         %
%                                                         %
% Autor : Alexandre Santos de la Vega                     %
% Datas : /2022-01/                                       %
%                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% limpeza do ambiente de trabalho
clear all
close all

% definicao da abscissa
n = -1.5:0.01:1.5;

% definicao da ordenada
y1 = (n);
```

```
y2 = (n.^2);
y3 = (n.^3);
y4 = (n.^4);

% criacao de uma tela de desenho (canvas)
figure (3)

% habilitacao da superposicao
hold on

% criacao de desenho
stem(n,y1,'b')
stem(n,y2,'g')
stem(n,y3,'r')
stem(n,y4,'k')

% criacao de legenda
legend('n^1','n^2','n^3','n^4', 'Location', 'southeast')

% criacao de labels
xlabel('n')
ylabel('y[n] = n^k  ,  k = 1:4')

% criacao de titulo
title('Superposição de gráficos discretos')

%
%EOF
%
```

Código 1.7: Exemplo de criação de subjanela ocupando vários espaços.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: Aula_3_Lab_Octave_Curvas_4.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                         %
% Titulo:                                                 %
% Exemplo de criação de subjanela ocupando vários espaços %
%                                                         %
%                                                         %
% Autor : Alexandre Santos de la Vega                     %
% Datas : /2022-01/                                       %
%                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% limpeza do ambiente de trabalho
clear all
close all

% definicao da abscissa
n = -1.5:0.01:1.5;
```

```
27
   % definicao da ordenada
29 y1 = (n);
   y2 = (n.^2);
31 y3 = (n.^3);
   y4 = (n.^4);
33
   % criacao de uma tela de desenho (canvas)
35 figure (4)

37 % criacao de subjanela na tela
   subplot(2,4,1)
39
   % criacao de desenho
41 stem(n,y1,'b')

43 % criacao de titulo
   title('Polinômios de grau ímpar')
45
   % criacao de labels
47 ylabel('y[n] = n^1')

49 % criacao de subjanela na tela
   subplot(2,4,4)
51
   % criacao de desenho
53 stem(n,y2,'g')

55 % criacao de titulo
   title('Polinômios de grau par')
57
   % criacao de labels
59 ylabel('y[n] = n^2')

61 % criacao de subjanela na tela
   subplot(2,4,5)
63
   % criacao de desenho
65 stem(n,y3,'r')

67 % criacao de labels
   ylabel('y[n] = n^3')
69 xlabel('n')

71 % criacao de subjanela na tela
   subplot(2,4,8)
73
   % criacao de desenho
75 stem(n,y4,'k')

77 % criacao de labels
   ylabel('y[n] = n^4')
79 xlabel('n')

81 % criacao de subjanela na tela
   subplot(2,4,[2:3 6:7])
83
   % habilitacao da superposicao
```

```
hold on

% criacao de desenho na tela
stem(n,y1,'b')
stem(n,y2,'g')
stem(n,y3,'r')
stem(n,y4,'k')

% criacao de legenda
legend('n^1','n^2','n^3','n^4', 'Location', 'southeast')

% criacao de labels
xlabel('n')
ylabel('y[n] = n^k  ,   k = 1:4')

% criacao de titulo
title('Superposição de gráficos discretos')


%
%EOF
%
```

Código 1.8: Exemplos de criação de curvas discretas e contínuas.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: Aula_3_Lab_Octave_Curvas_5.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                      %
% Titulo:                                              %
% Exemplo de criação:                                  %
%     de curva discreta,                               %
%     de curva contínua por meio de interpolador de ordem 0 %
%     e                                                %
%     de curva contínua por meio de interpolador de ordem 1 %
%                                                      %
%                                                      %
% Autor : Alexandre Santos de la Vega                  %
% Datas : /2022-01/                                    %
%                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% limpeza do ambiente de trabalho
clear all
close all

% definicao do periodo
Np = 40;

% definicao do indice
n  = 0:Np;
```

```
34 % definicao da frequencia discreta
   Omega = 2*pi / Np;
36
   % definicao da amplitude
38 A = 1;

40 % definicao da abscissa
   y = A * cos(Omega * n);
42
   % definicao do periodo de amostragem e do tempo
44 Ts = 1 / 44e3;
   t  = n * Ts;
46
   %
48 FigNbr = 4;

50 %
   FigNbr = FigNbr + 1;
52 figure (FigNbr)
   %
54 subplot(2,2,1)
   stem(n,y,'k')
56 title('Curva discreta')
   ylabel('y[n] = A cos (\Omega n + \Theta)')
58 xlabel('n')
   axis( [0 n(end) min(y) max(y) ])
60 %
   subplot(2,2,2)
62 stairs(t,y,'r')
   title( {'Curva produzida por interpolador de ordem 0 (ZOH)',
64         'com Fs = 44 kHz'} )
   ylabel('y(t) = ZOH \{ y[n] \}')
66 xlabel('t (s)')
   axis( [0 t(end) min(y) max(y) ])
68 %
   subplot(2,2,3)
70 plot(t,y,'b')
   title( {'Curva produzida por interpolador de ordem 1 (FOH)',
72         'com Fs = 44 kHz'} )
   ylabel('y(t) = FOH \{ y[n] \}')
74 xlabel('t (s)')
   axis( [0 t(end) min(y) max(y) ])
76 %
   subplot(2,2,4)
78 hold on
   stem(t,y,'k')
80 stairs(t,y,'r')
   plot(t,y,'b')
82 title('Superposição das curvas')
   %ylabel('y[n] = sin (\Omega n)')
84 xlabel('t (s)')
   axis( [0 t(end) min(y) max(y) ])
86


88 %
   %EOF
90 %
```

1.4 Representação gráfica tridimensional (curvas e superfícies)

1.4.1 Representação gráfica discreta de função bidimensional

- Um gráfico discreto de função bidimensional $z(x, y) = f(x, y)$ pode ser visto como uma sucessão de pontos $p = \{p_k\}$, formados por triplas $p_k = (x_k, y_k, z_k)$.
- Assim, o processo de se elaborar o gráfico pode ser definido como:
  - Montar os vetores $x = \{x_k\}$, $y = \{y_k\}$ e $z = \{z_k\}$.
  - Montar triplas $p_k = (x_k, y_k, z_k)$.
  - Definir um padrão representativo para um ponto em um espaço tridimensional.
  - Desenhar os pontos p_k no espaço tridimensional.

1.4.2 Representação gráfica contínua de função bidimensional

- Em um ambiente digital, não é possível traçar um gráfico contínuo.
- Porém, a partir de pontos do gráfico contínuo, pode-se obter uma aproximação interpolada.

1.4.3 Funções comumente utilizadas

- Funções relacionadas com gráficos:
  figure(), subplot(), axis(), hold on/off, title(), xlabel(), ylabel(), zlabel(), linspace(), logspace().
- Funções relacionadas com gráficos bidimensionais genéricos:
  help graph3d, stem3(), plot3(), surface(), surf(), surfc(), mesh(), meshc(), waterfall(), view().
- Funções relacionadas com gráficos bidimensionais especializados:
  help specgraph, bar3(), barh3(), pie3(), contour3(), quiver3().

1.4.4 Códigos

- O Código 1.9 apresenta um exemplo de criação de curva 3D. Relações entre exp(), sin() e cos(). Casos discreto e contínuo.
- O Código 1.10 apresenta um exemplo de criação de superfície discreta.

Código 1.9: Exemplo de criação de curva discreta 3D.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo original  : Aula_4_Lab_Octave_Curva_3D.m
% Arquivo atualizado: Exp_sin_cos_2021_07_05.m
%
% Arquivo: Aula_4_Lab_Octave_Exp_3D_Sin_Cos.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                    %
% Titulo:                            %
% Exemplo de criacao de curva 3D     %
%  - Relacoes: exp(), sin() e cos(). %
%  - Casos: continuo e discreto.     %
%                                    %
%                                    %
% Autor : Alexandre Santos de la Vega %
% Datas : /2021_07_05/               %
%                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% limpeza do ambiente de trabalho
clear all
close all

% definicao do periodo
Np = 40;

% definicao do indice
n = 0:Np;

% definicao da frequencia discreta
Omega = 2*pi / Np;

% definicao da abscissa
x = cos(Omega * n);

% definicao da ordenada
y = sin(Omega * n);

% definicao da altura
z = n;

% definicao do periodo de amostragem e do tempo
Ts = 1 / 44e3;
t = n * Ts;


%
FigNbr = 0;
black_val = 0;

%
FigNbr = FigNbr + 1;
figure(FigNbr)
%
%
subplot(2,2,1)
plot(x,y,"o")
axis square
title('Curva discreta 2D: e^{j (\Omega n)}')
ylabel('y[n] = sin (\Omega n)')
xlabel('x[n] = cos (\Omega n)')
%
% fundo preto
```

```
%set(gca,'color',[black_val,black_val,black_val])
%
subplot(2,2,2)
stem(n,y)
axis( [0 n(end) min(y) max(y) ])
title('Projecao no eixo Y')
ylabel('y[n] = sin (\Omega n)')
xlabel('n')
% fundo preto
%set(gca,'color',[black_val,black_val,black_val])
%
subplot(2,2,3)
stem(n,x)
axis square
axis( [0 n(end) min(y) max(y) ])
title('Projecao no eixo X')
ylabel('x[n] = cos (\Omega n)')
xlabel('n')
%
% fundo preto
%set(gca,'color',[black_val,black_val,black_val])
%
%view (AZIMUTH, ELEVATION)
view (90,90)

%
disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

%
FigNbr = FigNbr + 1;
figure(FigNbr)
%
subplot(2,2,1)
plot(x,y)
axis square
title('Curva continua 2D: e^{j (\omega t)}')
ylabel('y(t) = sin (\omega t)')
xlabel('x(t) = cos (\omega t)')
%
% fundo preto
%set(gca,'color',[black_val,black_val,black_val])
%
subplot(2,2,2)
plot(t,y)
axis( [0 t(end) min(y) max(y) ])
title('Projecao no eixo Y')
ylabel('y(t) = sin (\omega t)')
xlabel('t')
%
% fundo preto
%set(gca,'color',[black_val,black_val,black_val])
%
subplot(2,2,3)
plot(t,x)
axis square
axis( [0 t(end) min(y) max(y) ])
title('Projecao no eixo X')
```

```
128 ylabel('x(t) = cos (\omega t)')
    xlabel('t')
130 %
    % fundo preto
132 %set(gca,'color',[black_val,black_val,black_val])
    %
134 %view (AZIMUTH, ELEVATION)
    view (90,90)
136
    %
138 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
140 pause()

142 %
    FigNbr = FigNbr + 1;
144 figure(FigNbr)
    %
146 subplot(2,2,1)
    plot(x,y,"o")
148 axis square
    title('Projecao no plano X x Y')
150 ylabel('y[n] = sin (\Omega n)')
    xlabel('x[n] = cos (\Omega n)')
152 %
    % fundo preto
154 %set(gca,'color',[black_val,black_val,black_val])
    %
156 subplot(2,2,2)
    stem(n,y)
158 axis( [0 n(end) min(y) max(y) ])
    title('Projecao no plano Z x Y')
160 ylabel('y[n] = sin (\Omega n)')
    xlabel('z[n] = n')
162 %
    % fundo preto
164 %set(gca,'color',[black_val,black_val,black_val])
    %
166 subplot(2,2,3)
    stem(n,x)
168 axis square
    axis( [0 n(end) min(y) max(y) ])
170 title('Projecao no plano Z x X')
    ylabel('x[n] = cos (\Omega n)')
172 xlabel('z[n] = n')
    %
174 % fundo preto
    %set(gca,'color',[black_val,black_val,black_val])
176 %
    %view (AZIMUTH, ELEVATION)
178 view (90,90)
    %
180 subplot(2,2,4)
    stem3(x,y,z)
182 title('Curva discreta 3D: e^{j (\Omega n)}')
    zlabel('z[n] = n')
184 ylabel('y[n] = sin (\Omega n)')
    xlabel('x[n] = cos (\Omega n)')
186 %
```

```
% fundo preto
%set(gca,'color',[black_val,black_val,black_val])

%
disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

%
FigNbr = FigNbr + 1;
figure(FigNbr)
%
subplot(2,2,1)
plot(x,y)
axis square
title('Projecao no plano X x Y')
ylabel('y(t) = sin (\omega t)')
xlabel('x(t) = cos (\omega t)')
%
% fundo preto
%set(gca,'color',[black_val,black_val,black_val])
%
subplot(2,2,2)
plot(t,y)
axis( [0 t(end) min(y) max(y) ])
title('Projecao no plano Z x Y')
ylabel('y(t) = sin (\omega t)')
xlabel('z(t) = t')
%
% fundo preto
%set(gca,'color',[black_val,black_val,black_val])
%
subplot(2,2,3)
plot(t,x)
axis square
axis( [0 t(end) min(y) max(y) ])
title('Projecao no plano Z x X')
ylabel('x(t) = cos (\omega t)')
xlabel('z(t) = t')
%
% fundo preto
%set(gca,'color',[black_val,black_val,black_val])
%
%view (AZIMUTH, ELEVATION)
view (90,90)
%
subplot(2,2,4)
plot3(x,y,t)
grid on
title('Curva continua 3D: e^{j (\omega t)}')
zlabel('z(t) = t')
ylabel('y(t) = sin (\omega t)')
xlabel('x(t) = cos (\omega t)')
%
% fundo preto
%set(gca,'color',[black_val,black_val,black_val])

%
%EOF
```

```
%
```

Código 1.10: Exemplo de criação de superfície discreta.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: Aula_4_Lab_Octave_Superficies.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                            %
% Titulo:                                    %
% Exemplo de criacao de superficie discreta 3D %
%                                            %
%                                            %
% Autor : Alexandre Santos de la Vega        %
% Datas : /2022-01/                          %
%                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% limpeza do ambiente de trabalho
clear all
close all

%
x = -1:0.01:1;
y = x;
[X,Y] = meshgrid(x,y);

%
%z = sqrt(X) + sqrt(Y);
z = X.^2 + Y.^2;

%
FigNbr = 0;

%
FigNbr = FigNbr + 1;
figure(FigNbr)
%
surf(X,Y,z)

disp(' ')
disp('Pressione qualquer tecla para continuar...')
pause()

%
FigNbr = FigNbr + 1;
figure(FigNbr)
%
surfc(X,Y,z)

disp(' ')
disp('Pressione qualquer tecla para continuar...')
```

```
54 pause()

56 %
   FigNbr = FigNbr + 1;
58 figure(FigNbr)
   %
60 mesh(X,Y,z)

62 disp(' ')
   disp('Pressione qualquer tecla para continuar...')
64 pause()

66 %
   FigNbr = FigNbr + 1;
68 figure(FigNbr)
   %
70 meshc(X,Y,z)

72 disp(' ')
   disp('Pressione qualquer tecla para continuar...')
74 pause()

76 %
   FigNbr = FigNbr + 1;
78 figure(FigNbr)
   %
80 meshz(X,Y,z)

82 disp(' ')
   disp('Pressione qualquer tecla para continuar...')
84 pause()

86 %
   FigNbr = FigNbr + 1;
88 figure(FigNbr)
   %
90 waterfall(X,Y,z)

92 %
   % EOF
94 %
```

1.5 Representação gráfica tridimensional (imagens)

1.5.1 Representação gráfica discreta de função bidimensional

- Uma imagem discreta pode ser interpretada como um gráfico discreto de uma função bidimensional.
- Um gráfico discreto de função bidimensional $z(x, y) = f(x, y)$ pode ser visto como uma sucessão de pontos $p = \{p_k\}$, formados por triplas $p_k = (x_k, y_k, z_k)$.
- Podem ser destacadas duas diferenças básicas entre o gráfico de uma superfície e o gráfico de uma imagem:
  - Na superfície, o valor de z_k representa a distância do ponto p_k ao plano (x, y), gerando objetos tridimensionais. Por sua vez, na imagem, o valor de z_k representa a intensidade de cor do ponto p_k na posição (x_k, y_k), gerando objetos bidimensionais.
  - Na superfície, cada ponto é representado de forma adimensional. Para se obter uma aproximação de uma superfície contínua por uma superfície discreta, deve-se realizar alguma forma de interpolação entre seus pontos. Por sua vez, na imagem, cada ponto já é naturalmente associado a uma área finita bidimensional, preenchida com alguma cor, em torno da posição (x_k, y_k) do ponto p_k.
- Cada ponto p_k de uma imagem representa o seu elemento básico constituinte e é denominado de *pixel* (*picture element*).
- A quantidade de *bits* usados para armazenar informação sobre cada *pixel* é denominada de *bit depth* (*bits per pixel*).
- Assim, o processo de se elaborar a imagem pode ser definido como:
  - Montar os vetores $x = \{x_k\}$, $y = \{y_k\}$ e $z = \{z_k\}$.
  - Montar triplas $p_k = (x_k, y_k, z_k)$.
  - Definir um padrão representativo para um *pixel*, representado por um ponto em um espaço bidimensional. Normalmente, costuma-se preencher uma área em torno de cada ponto desenhado.
  - Desenhar os pontos p_k (*pixels*) no espaço bidimensional.

1.5.2 Representações de imagens no aplicativo

- Formatos padrões de armazenamento de imagens suportados:
  - BMP (*Microsoft Windows Bitmap*).
  - HDF (*Hierarchical Data Format*).
  - JPEG (*Joint Photographic Experts Group*).
  - PCX (*Paintbrush*).
  - PNG (*Portable Network Graphics*).
  - TIFF (*Tagged Image File Format*).
  - XWD (*X Window Dump*).
- Tipo de dados usados para representar imagens: uint8, uint16 e double.

1.5.3 Estruturas de dados para imagens no aplicativo

Imagem indexada

- Utiliza duas matrizes por imagem: matrix de *pixels* e matriz de mapa de cores.
- Matriz de *pixels* (LxC)
  - Contém valores inteiros, que servem de índices para acesso ao mapa de cores.
  - Para dados uint8/16, a faixa de índices é [0;Lm-1].
  - Para dados double, a faixa de índices é [1;Lm].
  - O aplicativo não realiza operações matemáticas sobre dados uint8 e uint16. Para isso, podem-se empregar as seguintes conversões:
    - uint8 $\longrightarrow$ double: M64 = double(M8) + 1.
    - uint16 $\longrightarrow$ double: M64 = double(M16) + 1.
    - double $\longrightarrow$ uint8 : M8 = uint8 (round(M64 - 1)).
    - double $\longrightarrow$ uint16: M16 = uint16(round(M64 - 1)).
- Matriz de mapa de cores (Lmx3)
  - Contém valores double, na faixa [0;1].
  - As colunas 1 a 3 representam, respectivamente, as intensidades das cores vermelho (*red* ou R), verde (*green* ou G) e azul (*blue* ou B).
  - A linha (0,0,0) representa a menor intensidade possível, o que significa a "cor" preto.
  - A linha (1,1,1) representa a maior intensidade possível, o que significa a "cor" branco.

Imagem de intensidade ou escala em tons de cinza ou *grayscale*

- Utiliza uma matriz por imagem: matriz de *pixels*.
- Matriz de *pixels* (LxC): contém valores de intensidade, que se encontram dentro de alguma faixa.
- O aplicativo manipula a imagem de intensidade como se fosse uma imagem indexada.
- A matriz de intensidade é utilizada como se fosse uma matriz de índices, associada a um mapa de cores especificado.
- Normalmente, é utilizado um mapa com escala de tons de cinza (*grayscale*).
- Os valores de intensidade são linearmente escalados para gerar os índices de acesso ao mapa de cores.
- Os valores de intensidade podem ser uint8, uint16 ou double.
- Valores típicos de intensidade:
  - Para dados uint8 , a faixa é [0;255].
  - Para dados uint16, a faixa de índices é [0;65535].
  - Para dados double, a faixa de índices é [0;1].

Imagem RGB ou *Truecolor*

- Utiliza uma matriz por imagem: matriz de *pixels*.
- Matriz de *pixels* (LxCx3): contém valores de intensidade, para cada uma das cores básicas (RGB).
- Os valores de intensidade podem ser uint8, uint16 ou double.
- Valores típicos de intensidade:
  - Para dados uint8 , a faixa é [0;255].
  - Para dados uint16, a faixa de índices é [0;65535].
  - Para dados double, a faixa de índices é [0;1].

1.5.4 Amostragem e exibição das imagens

- Na amostragem de imagens, quanto menor for o intervalo de amostragem (resolução) utilizado na geração da imagem discreta, maior será o número de pontos p_k (*pixels*) obtidos.
- Na exibição das imagens armazenadas em matrizes, considera-se que o ponto p_k (*pixel*) ocupa uma determinada área em torno da posição (x_k, y_k), a qual é preenchida com a cor definida pelo valor z_k da matriz. Quanto menor for a área ocupada por cada *pixel*, maior será a densidade superficial de pontos. Uma unidade comumente utilizada para expressar a densidade superficial de pontos é "Pontos Por Polegada" ou *Dots Per Inch* ou DPI.
- Levando-se em consideração tanto a resolução usada na amostragem de uma imagem quanto a densidade superficial de pontos usada na sua reprodução, pode-se dizer que, independentemente do tamanho original da imagem, quanto menor for o valor da resolução e quanto maior for a densidade superficial, maior será a qualidade visual da reprodução da imagem.
- Na tentativa de se ocupar menos espaço no armazenamento, pode-se aplicar a operação de *downsampling*. Por exemplo:
  - Aplicando-se um *downsampling* de 2 em x, elimina-se uma coluna entre cada duas antigas colunas.
  - Aplicando-se um *downsampling* de 2 em y, elimina-se uma linha entre cada duas antigas linhas.
- Por outro lado, na tentativa de se melhorar a resolução de uma imagem armazenada, diminuindo-se o seu valor, pode-se aplicar a operação de *upsampling*, seguida de algum método de interpolação. Por exemplo:
  - Aplicando-se um *upsampling* de 2 em x, abre-se uma nova coluna entre cada duas antigas colunas e novos valores devem ser calculados.
  - Aplicando-se um *upsampling* de 2 em y, abre-se uma nova linha entre cada duas antigas linhas e novos valores devem ser calculados.
- Um método de interpolação simples de se implementar é a interpolação linear, que, para um *upsampling* de 2, resume-se ao cálculo de uma média aritmética de dois valores.

1.5.5 Funções comumente utilizadas

- Funções relacionadas com gráficos:
  figure(), subplot(), axis(), hold on/off, title(), xlabel(), ylabel(), zlabel(), linspace(), logspace().
- Funções relacionadas com imagens:
  image(), imagesc(), colormap(), colorbar(), daspect(), imread(), imwrite(), iminfo(), ind2rgb().
- Mapas de cores padrões:
  - 1 cor: white.
  - 16 cores: vga.
  - 64 cores: hsv, hot, gray, bone, copper, pink, flag, lines, colorcube, jet, prism, cool, autumn, spring, winter, summer.

1.5.6 Códigos

- O Código 1.11 apresenta exemplos básicos na geração de imagens em tons de cinza.
- O Código 1.12 apresenta um exemplo de geração de imagem baseada em mapa de cor, a partir de uma matriz.

Código 1.11: Exemplos básicos na geração de imagens em tons de cinza.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: Aula_5_Lab_Octave_Imagens_Basics.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                  %
% Titulo:                                          %
% Exemplo de gráfico discreto de função bidimensional  %
%   - Imagem                                       %
%                                                  %
% Exemplos básicos na geração de imagens em tons de cinza %
%                                                  %
%                                                  %
% Autor : Alexandre Santos de la Vega              %
% Datas : /2022-01/                                %
%                                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
% ==================================================
%

% limpeza do ambiente de trabalho
%
clear all
```

```
30 close all

32 %
   % ========================================================
34 %

36 % define niveis de branco e preto
   c_black = 0;
38 c_white = 64;

40 % inicializa o numero do canvas
   FigNbr = 0;
42
   %
44 % ========================================================
   %
46
   % desenha 1 pixel
48 %
   FigNbr = FigNbr+1 ;
50 figure(FigNbr)
   %
52 map = gray;
   colormap(map);
54 %
   m_image = [c_black];
56 %
   image(m_image)
58 f = size(m_image);
   axis([0 f(2)+1 0 f(1)+1])
60
   disp(' ')
62 disp('Pressione qualquer tecla para continuar...')
   pause()
64
   axis square
66 %axis image

68 disp(' ')
   disp('Pressione qualquer tecla para continuar...')
70 pause()

72 %
   % ========================================================
74 %

76 % desenha alguns pixels na forma de uma matriz
   %
78 FigNbr = FigNbr+1 ;
   figure(FigNbr)
80 %
   map = gray;
82 colormap(map);
   %
84 m_image = c_white * ones(3,3);
   m_image([1,3],[1,3]) = c_black;
86 %
   image(m_image)
88 f = size(m_image);
```

```
    axis([0 f(2)+1 0 f(1)+1])
90
    disp(' ')
92  disp('Pressione qualquer tecla para continuar...')
    pause()
94
    axis square
96  %axis image

98  disp(' ')
    disp('Pressione qualquer tecla para continuar...')
100 pause()

102 %
    %=======================================================
104 %

106 % desenha alguns pixels na forma de uma matriz
    %
108 FigNbr = FigNbr+1 ;
    figure(FigNbr)
110 %
    map = gray;
112 colormap(map);
    %
114 m_image = c_white * ones(5,5);
    m_image(1:2:5,1:2:5) = c_black;
116 %
    image(m_image)
118 f = size(m_image);
    axis([0 f(2)+1 0 f(1)+1])
120
    disp(' ')
122 disp('Pressione qualquer tecla para continuar...')
    pause()
124
    axis square
126 %axis image

128 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
130 pause()

132 %
    %=======================================================
134 %

136 % desenha uma cruz
    %
138 FigNbr = FigNbr+1 ;
    figure(FigNbr)
140 %
    map = gray;
142 colormap(map);
    %
144 m_image = c_white * ones(3,3);
    m_image(1:2:5,1:2:5) = c_black;
146 %
    image(m_image)
```

```
148 f = size(m_image);
    axis([0 f(2)+1 0 f(1)+1])
150
    disp(' ')
152 disp('Pressione qualquer tecla para continuar...')
    pause()
154
    axis square
156 %axis image

158 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
160 pause()

162 %
    % ==========================================================
164 %

166 % desenha sinais de + e - dentro de um frame
    %
168 FigNbr = FigNbr+1 ;
    figure(FigNbr)
170 %
    map = gray;
172 colormap(map);
    %
174 m_image = c_white * ones(15,15);
    %
176 m_image(1       , 1:end) = c_black;
    m_image(1:end  , end   ) = c_black;
178 m_image(end     , 1:end) = c_black;
    m_image(1:end  , 1     ) = c_black;
180 %
    m_image(1+3       , 1+(2:4)) = c_black;
182 m_image(1+(2:4)  , 1+3     ) = c_black;
    %
184 m_image(end-3     , end-(2:4)) = c_black;
    %
186 image(m_image)
    f = size(m_image);
188 axis([0 f(2)+1 0 f(1)+1])

190 disp(' ')
    disp('Pressione qualquer tecla para continuar...')
192 pause()

194 axis square
    %axis image
196
    disp(' ')
198 disp('Pressione qualquer tecla para continuar...')
    pause()
200
    %
202 %EOF
    %
```

Código 1.12: Exemplo de geração de imagem baseada em mapa de cor, a partir de uma matriz.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: heatmap_propagacao_modelo_de_friis.m.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                      %
% Arquivo:                             %
%   - Heatmap, com definição de cMap.  %
%   - Modelo de espaço livre de Friis. %
%   - Gráfico de Perda.                %
%   - Identificação de posição de TX.  %
%   - Identificação de campo distante. %
%                                      %
%                                      %
% Autor: Alexandre Santos de la Vega   %
%                                      %
% Data: /2024-01/                      %
%                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% ====================================================
%

% limpeza do ambiente de trabalho
%
clear all
close all

%
% ====================================================
%

% define parâmetros comuns

% velocidade da luz no vácuo
c =  3e8 ; % m/s

% frequência da antena TX
Ft = 1e6 ; % Hz

% comprimento de onda
lambda = (c / Ft) ; % m

% resolução da amostragem
y_res = .25 * lambda ; % m
x_res = .25 * lambda ; % m

%
% ================================================
%

% define ambiente de propagação

```

```
% limite contínuo
y_lim = 8e3 ; % max y  (m)
x_lim = 12e3 ; % max x  (m)

% limite discreto
r_size = fix(y_lim / y_res) + 1 ; % max linha
c_size = fix(x_lim / x_res) + 1 ; % max coluna

% matriz de distâncias em relação à antena TX
m_dist = zeros(r_size, c_size) ; % ambiente discreto

%
% ========================================================
%

% define posição da antena TX

% posição contínua
%
% for testing...
y_tx_pos = y_lim/2 ; % m
x_tx_pos = x_lim/2 ; % m
%
% desired...
%y_tx_pos = 10 ; % m
%x_tx_pos = 20 ; % m

% posição discreta
r_tx_pos = fix(y_tx_pos / y_res) ;
c_tx_pos = fix(x_tx_pos / x_res) ;

%
% ========================================================
%

% realiza cálculo das distâncias
%
for row=1:r_size
  for col=1:c_size
    m_dist (row,col) = sqrt ( ( (row*y_res) - y_tx_pos )^2
                              +
                              ( (col*x_res) - x_tx_pos )^2 ) ;
  endfor
endfor
%
clear row col;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% realiza cálculo das variáveis (potência, perda, etc.)

% define parâmetros
Pt = .25 ; % W
Gt = 1 ;
Gr = 1 ;

% calcula fator de perda
mAtr = ( lambda ./ (4 * pi * m_dist) ) .^ 2 ;
```

```
117 % calcula a potência recebida
    %
119 % Modelo de Free Space de Friis
    Pr = Pt * Gt * Gr * mAtr ;
121
    % calcula a perda
123 z = (Pt ./ Pr) ;
    %z_db = 10 * log10(z);
125
    %
127 % ================================================
    %
129
    %%%%%%%%%%%%%%%%%%%%
131
    % gera gráficos
133
    %%%%%%%%%%%%%%%%%%%%
135
    % cria canvas
137 %
    figure
139
    %%%%%%%%%%%%%%%%%%%%
141
    % cria um colormap green2red
143 %
    %   mecanismo 1
145 %
    %green = [0 1 0] ;
147 %red = [1 0 0] ;
    %n = 256 ;
149 %Map = [linspace(green(1),red(1),n)'
    %        linspace(green(2),red(2),n)'
151 %        linspace(green(3),red(3),n)'] ;
    %
153 %   mecanismo 2
    %
155 cMap = interp1([0;1],[0 1 0; 1 0 0],linspace(0,1,256)) ;

157 % define o mapa de cores
    %
159 colormap(cMap) ;

161 %%%%%%%%%%%%%%%%%%%%

163 % define valores dos eixos:
    %   discretos (0) ou contínuos (1)
165 %
    axis_choice = 0 ;
167 axis_choice = 1 ;

169 %%%%%%%%%%%%%%%%%%%%

171 % define os limites de visualizacao
    %
173 if (axis_choice==0)
      % limites discretos
175   x_lims = [0 , (c_size-1)] ;
```

```
    y_lims = [0 , (r_size-1)] ;
177 else
    % limites contínuos
179   x_lims = [(x_res * 0) , (x_res * (c_size - 1))] ;
    y_lims = [(y_res * 0) , (y_res * (r_size - 1))] ;
181 endif

183 %%%%%%%%%%%%%%%%%%%%

185 % desenha a imagem da matriz no canvas,
    % usando o mapa de cores escolhido.
187 %
    function_choice = 1 ; % imagesc()
189 %function_choice = 2 ; % image()
    %function_choice = 3 ; $ imshow()
191 %
    if (function_choice==1)
193   imagesc(x_lims,y_lims,z) ;
    elseif (function_choice==2)
195   image(x_lims,y_lims,z) ;
    else
197 %  imshow(z,"colormap",cMap,"xdata",x_lims,"ydata",y_lims);
    imshow(z,cMap) ;
199   endif
    %
201 % retira a ordem "reverse" do eixo y...
    set(gca,"YDir","normal") ;
203
    % define aspect ratio
205 daspect ([1 1])

207 %%%%%%%%%%%%%%%%%%%%

209 % coloca identificadores
    %
211 tit_str = cellstr(
      ['Heatmap do modelo de espaço livre de Friis';
213    'Gráfico da perda: L = (Pt / Pr)';
       'Identificações: posição do TX e início do campo distante.';
215    'Parâmetros: c = ', num2str(c), ...
         ' m/s  ;  f = ', num2str(Ft), ...
217      ' Hz  ;  \lambda = ', num2str(lambda), ' m  ; ';
       'Pt = ', num2str(Pt), ...
219      ' W  ;  Gt = ', num2str(Gt), ...
         '  ;  Gr = ', num2str(Gr), ' .';
221    'Resoluções: x =  ', num2str(x_res), ...
         ' m  ;  y = ', num2str(y_res), ' m .']) ;
223 title(tit_str)
    %
225 if (axis_choice==0)
      % limites discretos
227   %
      ylabel('Linha')
229   xlabel('Coluna')
    else
231   % limites contínuos
      %
233   ylabel('Dimensão y  (m)')
      xlabel('Dimensão x  (m)')
```

```
235 endif

237 %%%%%%%%%%%%%%%%%%%%

239 % desenha uma barra de cores,
    % usando o mapa de cores escolhido.
241 %% Obs.: colocar depois de desenhar a imagem...
    %
243 colorbar ("westoutside") ;

245 %%%%%%%%%%%%%%%%%%%%

247 % mantém os gráficos anteriores
    %
249 hold on

251 %%%%%%%%%%%%%%%%%%%%

253 % desenha um símbolo para a antena TX
    %
255 if (axis_choice==0)
      % limites discretos
257   %
      plot(c_tx_pos, r_tx_pos, "k*")
259 else
      % limites contínuos
261   %
      x_tx_pos = ((c_tx_pos-0) * x_res) ;
263   y_tx_pos = ((r_tx_pos-0) * y_res) ;
      %
265   plot(x_tx_pos, y_tx_pos, "k*")
    endif
267
    %%%%%%%%%%%%%%%%%%%%
269
    % desenha um círculo
271 % no início de campo distante de Friis
    %
273 if (axis_choice==0)
      % limites discretos
275   %
      r_disc = 10*lambda;
277   theta = 0:(pi/360):(2*pi);
      %
279   c_cd = c_tx_pos + fix(r_disc / x_res) * cos(theta) ;
      r_cd = r_tx_pos + fix(r_disc / y_res) * sin(theta) ;
281   %
      plot(c_cd , r_cd , "k")
283 else
      % limites contínuos
285   %
      r_cont = 10*lambda;
287   theta = 0:(pi/360):(2*pi);
      %
289   x_cd = x_tx_pos + r_cont * cos(theta) ;
      y_cd = y_tx_pos + r_cont * sin(theta) ;
291   %
      plot(x_cd , y_cd , "k")
293 endif
```

```
295 %%%%%%%%%%%%%%%%%%%%
297 %
    % ========================================================
299 %

301 %
    % EOF
303 %
```

1.6 Simulação de uma equação de diferença

1.6.1 Equação de diferença

- Uma equação de diferença, linear, com coeficientes constantes e causal, é definida por

$$a_0 y[n] + a_1 y[n-1] + \ldots + a_N y[n-N] = b_0 x[n] + b_1 x[n-1] + \ldots + b_L x[n-L] \ . \quad (1.1)$$

- Considerando-se $x[n] = 0$, para $n < 0$, e as condições iniciais $y_{CI} = \{y[-1], ..., y[-N]\}$, pode-se calcular o valor de $y[n]$, para $n \geq 0$, por meio de um processo iterativo, tal como:

$$y[n] = -\frac{a_1}{a_0} y[n-1] - \ldots - \frac{a_N}{a_0} y[n-N] + \frac{b_0}{a_0} x[n] + \frac{b_1}{a_0} x[n-1] + \ldots + \frac{b_L}{a_0} x[n-L] \ . \quad (1.2)$$

- Considerando-se os diversos resultados possivelmente obtidos por manipulações algébricas da Equação (1.1), cada um deles podem ser considerado um algoritmo diferente para o cálculo de y[n].

1.6.2 Simulação de uma equação de diferença no aplicativo

- A função $filter(\cdot)$ realiza a simulação da equação de diferença definida na Equação (1.1).

- Na sua forma mais simples, dada por

$$y = filter(b, a, x) \ ,$$

  a função calcula o vetor

$$y = \{y[0], y[1], \ \ldots \ , y[X]\} \ ,$$

  dados os vetores

$$b = \{b_0, b_1, \ \ldots \ , b_L\} \ ,$$

$$a = \{a_0, a_1, \ \ldots \ , a_N\}$$

  e

$$x = \{x[0], x[1], \ \ldots \ , x[X]\} \ , \ X \geq 0 \ .$$

- No caso de $a_0 \neq 1$, a função normaliza todos os coeficientes por a_0, assim como na Equação (1.2).

1.6.3 Códigos

- O Código 1.13 apresenta um exemplo de simulação de uma equação de diferença.

Código 1.13: Exemplo de simulação de uma equação de diferença.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: Aula_7_Lab_Octave_ED_e_filter.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
% Titulo:                                                                       %
% Exemplo de simulação de uma equação de diferença                              %
%                                                                               %
%                                                                               %
% Autor : Alexandre Santos de la Vega                                           %
% Datas : /2022-01/                                                             %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
% ================================================================
%

% limpeza do ambiente de trabalho
%
clear all
close all

%
% ================================================================
%

% define os coeficientes da equacao de diferenca
%
% b = [ b0 b1 b2 ]
b = [ 0.435390588649551 0.507111243808470 0.435390588649552 ];
%
% a = [ a0 a1 a2]
a = [ 1.000000000000000 0.113933557377612 0.263958863729961 ];

% inicializa o numero do canvas
FigNbr = 0;

%
% ================================================================
%

% define um impulso discreto unitario
%
Nud = 21;
%
unit_delta = zeros(1,Nud);
unit_delta(1) = 1;
```

```
55 % define um degrau discreto unitario
   %
57 Nus = 31;
   %
59 unit_step = ones(1,Nus);

61 % define um gate retangular unitario deslocado
   %
63 Nurg = 61;
   unit_ret_gate = zeros (1,Nurg);
65 unit_ret_gate(10:20) =  1;
   unit_ret_gate(40:50) = -1;
67
   %
69 % ================================================
   %
71
   % calcula saida para
73 % um impulso discreto unitario
   %
75 y_ud = filter(b,a,unit_delta);

77 % calcula saida para
   % um degrau discreto unitario
79 %
   y_us = filter(b,a,unit_step);
81
   % calcula saida para
83 % um gate retangular unitario deslocado
   %
85 y_urg = filter(b,a,unit_ret_gate);

87 %
   % ================================================
89 %

91 % desenha graficos das respostas
   %
93 FigNbr = FigNbr+1 ;
   figure(FigNbr)
95 %
   %
97 subplot(3,2,1)
   stem( (0:(Nud-1)) , unit_delta )
99 ylabel('y_{ud} [n]')
   title('Impulso discreto unitário')
101 v = axis;
   v(1) = 0;          % x_min
103 v(2) = (Nurg-1);  % x_max
   v(3) = -1.5;       % y_min
105 v(4) =  1.5;       % y_max
   axis( [ v(1) v(2) v(3) v(4) ] )
107 %
   subplot(3,2,3)
109 stem( (0:(Nus-1)) , unit_step )
   ylabel('y_{us} [n]')
111 title('Degrau discreto unitário')
   v = axis;
```

```
113 v(1) = 0;           % x_min
    v(2) = (Nurg-1);    % x_max
115 v(3) = -1.5;        % y_min
    v(4) =  1.5;        % y_max
117 axis( [ v(1) v(2) v(3) v(4) ] )
    %
119 subplot(3,2,5)
    stem( (0:(Nurg-1)) , unit_ret_gate)
121 ylabel('y_{urg} [n]')
    title('Gate retangular unitário deslocado')
123 v = axis;
    v(1) = 0;           % x_min
125 v(2) = (Nurg-1);    % x_max
    v(3) = -1.5;        % y_min
127 v(4) =  1.5;        % y_max
    axis( [ v(1) v(2) v(3) v(4) ] )
129 %
    xlabel('n')
131 %
    %
133 subplot(3,2,2)
    stem( (0:(Nud-1)) , y_ud )
135 ylabel('y_{ud} [n]')
    title('Resposta ao impulso discreto unitário')
137 v = axis;
    v(1) = 0;           % x_min
139 v(2) = (Nurg-1);    % x_max
    v(3) = -1.5;        % y_min
141 v(4) =  1.5;        % y_max
    axis( [ v(1) v(2) v(3) v(4) ] )
143 %
    subplot(3,2,4)
145 stem( (0:(Nus-1)) , y_us )
    ylabel('y_{us} [n]')
147 title('Resposta ao degrau discreto unitário')
    v = axis;
149 v(1) = 0;           % x_min
    v(2) = (Nurg-1);    % x_max
151 v(3) = -1.5;        % y_min
    v(4) =  1.5;        % y_max
153 axis( [ v(1) v(2) v(3) v(4) ] )
    %
155 subplot(3,2,6)
    stem( (0:(Nurg-1)) , y_urg)
157 ylabel('y_{urg} [n]')
    title('Resposta ao gate retangular unitário deslocado')
159 v = axis;
    v(1) = 0;           % x_min
161 v(2) = (Nurg-1);    % x_max
    v(3) = -1.5;        % y_min
163 v(4) =  1.5;        % y_max
    axis( [ v(1) v(2) v(3) v(4) ] )
165 %
    xlabel('n')
167
    %
169 %EOF
    %
```

1.7 Exercícios propostos

1.7.1 Vetores: construção e manipulação

1. Crie um vetor linha contendo os valores pares contidos na faixa $n = [1; 16]$.
2. Crie um vetor coluna contendo os valores ímpares contidos na faixa $n = [2; 22]$.
3. Crie um vetor linha contendo os valores pares contidos na faixa $n = [21; 53]$, seguidos dos valores ímpares contidos na faixa $n = [72; 92]$.
4. Crie um vetor linha contendo os valores $v = \{33, 42, 97, 53, 64, 75\}$.
5. Crie um vetor linha contendo os valores $v = \{75, 64, 53, 97, 42, 33\}$, a partir de um dado vetor linha contendo os valores $v = \{33, 42, 97, 53, 64, 75\}$.
6. Crie um vetor coluna contendo os valores $v = \{75, 64, 53, 97, 42, 33\}$.
7. Crie um vetor coluna contendo os valores $v = \{33, 42, 97, 53, 64, 75\}$, a partir de um dado vetor coluna contendo os valores $v = \{75, 64, 53, 97, 42, 33\}$.
8. Crie um vetor coluna contendo os valores $v = \{75, 64, 53, 97, 42, 33\}$, a partir de um dado vetor linha contendo os valores $v = \{33, 42, 97, 53, 64, 75\}$.
9. Crie um vetor linha contendo os valores $v = \{75, 64, 53, 97, 42, 33\}$, a partir de um dado vetor coluna contendo os valores $v = \{33, 42, 97, 53, 64, 75\}$.
10. Crie um vetor linha contendo os valores $v = \{43, 21, 10, 65, 54, 32\}$, a partir de um dado vetor linha contendo os valores $v = \{10, 21, 32, 43, 54, 65\}$.
11. Crie um vetor linha contendo os valores $v = \{10, 21, 0, 0, 54, 65\}$, a partir de um dado vetor linha contendo os valores $v = \{10, 21, 32, 43, 54, 65\}$.
12. Crie um vetor linha contendo os valores $v = \{10, 0, 32, 0, 0, 65\}$, a partir de um dado vetor linha contendo os valores $v = \{10, 21, 32, 43, 54, 65\}$.
13. Crie um vetor linha contendo os valores $v = \{10, 21, 54, 65\}$, a partir de um dado vetor linha contendo os valores $v = \{10, 21, 32, 43, 54, 65\}$.
14. Crie um vetor linha contendo os valores $v = \{10, 32, 65\}$, a partir de um dado vetor linha contendo os valores $v = \{10, 21, 32, 43, 54, 65\}$.

1.7.2 Matrizes: construção e manipulação

1. Crie a matriz $M = \begin{bmatrix} 8 & 2.6 & 39 \\ 104 & 1.5 & -17 \\ 0.9 & 212 & 48 \end{bmatrix}$.

2. Crie a matriz $M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 20 & 21 & 22 & 23 \\ 104 & 105 & 106 & 107 \end{bmatrix}$.

3. Crie a matriz $M = \begin{bmatrix} 1 & 2 & 4 & 8 & 16 \\ 3 & 6 & 12 & 24 & 48 \\ 9 & 18 & 36 & 72 & 144 \\ 27 & 54 & 108 & 216 & 432 \end{bmatrix}$.

4. Crie a matriz $M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 48 \end{bmatrix}$.

5. Crie a matriz $M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$ a partir das seguintes submatrizes:
   $m_1 = \begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix}$, $m_2 = \begin{bmatrix} 10 & 9 \\ 14 & 13 \end{bmatrix}$, $m_3 = \begin{bmatrix} 7 & 8 \\ 3 & 4 \end{bmatrix}$ e $m_4 = \begin{bmatrix} 11 & 15 \\ 12 & 16 \end{bmatrix}$.

6. Crie a matriz $M = \begin{bmatrix} 1 & 7 & 31 & 22 & 93 \\ 48 & 16 & 72 & 0.3 & 1 \\ 53 & 25 & -11 & 6 & 19 \\ 62 & 14 & 108 & -31 & 88 \end{bmatrix}$ a partir das seguintes submatrizes:
   $v_1 = [1, 7, 31, 22, 93]$, $v_2 = [88, -31, 108, 14, 62]$, $v_3 = [72, -11]$, $m_4 = \begin{bmatrix} 16 & 25 \\ 48 & 53 \end{bmatrix}$ e
   $m_5 = \begin{bmatrix} 0.3 & 6 \\ 1 & 19 \end{bmatrix}$.

7. Crie a matriz $M = \begin{bmatrix} 1 & 7 & 31 & 22 & 93 & 44 \\ 48 & 16 & 72 & 0.3 & 1 & 58 \\ 53 & 25 & -11 & 6 & 19 & 39 \\ 62 & 14 & 108 & -31 & 88 & 17 \end{bmatrix}$ a partir das seguintes submatrizes:
   $v_1 = [1, 48, 53, 62]$, $v_2 = [88, 93, 1, 19]$, $v_3 = [17, 39]$, $v_4 = [93, 44, 22]$, $v_5 = [48, 16, 72, 0.3, 1, 58]$,
   $m_6 = \begin{bmatrix} 7 & 16 \\ 31 & 72 \end{bmatrix}$ e $m_7 = \begin{bmatrix} 14 & 108 & -31 & 88 \\ 25 & -11 & 6 & 19 \end{bmatrix}$.

8. Crie a matriz $M_2 = \begin{bmatrix} 1 & 6 & 11 & 16 \\ 2 & 7 & 12 & 17 \\ 3 & 8 & 13 & 18 \\ 4 & 9 & 14 & 19 \\ 5 & 10 & 15 & 20 \end{bmatrix}$ a partir da matriz $M_1 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \end{bmatrix}$.

9. Crie a matriz $M_2 = \begin{bmatrix} 5 & 4 & 3 & 2 & 1 \\ 10 & 9 & 8 & 7 & 6 \\ 15 & 14 & 13 & 12 & 11 \\ 20 & 19 & 18 & 17 & 16 \end{bmatrix}$ a partir da matriz $M_1 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \end{bmatrix}$.

10. Crie a matriz $M_2 = \begin{bmatrix} 16 & 17 & 18 & 19 & 20 \\ 11 & 12 & 13 & 14 & 15 \\ 6 & 7 & 8 & 9 & 10 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}$ a partir da matriz $M_1 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \end{bmatrix}$.

11. Crie a matriz $M_2 = \begin{bmatrix} 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}$ a partir da matriz $M_1 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \end{bmatrix}$.

12. Crie a matriz $M_2 = \begin{bmatrix} 16 & 17 & 18 & 19 & 20 \\ 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{bmatrix}$ a partir da matriz $M_1 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \end{bmatrix}$.

13. Crie a matriz $M_2 = \begin{bmatrix} 4 & 5 & 1 & 2 & 3 \\ 9 & 10 & 6 & 7 & 8 \\ 14 & 15 & 11 & 12 & 13 \\ 19 & 20 & 16 & 17 & 18 \end{bmatrix}$ a partir da matriz $M_1 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \end{bmatrix}$.

14. Crie a matriz $M_2 = \begin{bmatrix} 3 & 4 & 5 & 1 & 2 \\ 8 & 9 & 10 & 6 & 7 \\ 13 & 14 & 15 & 11 & 12 \\ 18 & 19 & 20 & 16 & 17 \end{bmatrix}$ a partir da matriz $M_1 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \end{bmatrix}$.

1.7.3 Matrizes: cálculos

1. Dadas as matrizes $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ e $B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$, compare os resultados das seguintes operações:

   (a) $A * B$, $B * A$, $A. * B$, $B. * A$.

   (b) A/B, $A\backslash B$, B/A, $B\backslash A$,
   $A * inv(B)$, $inv(A) * B$, $B * inv(A)$, $inv(B) * A$,
   $(B'\backslash A')'$, $(A'\backslash B')'$,
   $inv(B') * A'$, $inv(A') * B'$,
   $A./B$, $A.\backslash B$, $B./A$, $B.\backslash A$.

2. Mostre como usar a função $find(\cdot)$ para encontrar elementos genéricos em uma matriz.

3. Mostre como usar a função $sign(\cdot)$ para encontrar elementos genéricos em uma matriz.

1.7.4 Modelagem matricial

1. Escreva a Equação (1.3) na forma matricial da Equação (1.4), definindo a matriz $\boldsymbol{D_N}$. Em seguida, para $W_N = e^{-j\left(\frac{2\pi}{N}\right)}$ e um dado vetor $\boldsymbol{x}[n] = [\ x[0]\ x[1]\ \cdots\ x[N-1]\ ]$, calcule matriz $\boldsymbol{D_N}$ e o vetor $\boldsymbol{X}[k] = [\ X[0]\ X[1]\ \cdots\ X[N-1]\ ]$ usando a equação matricial proposta.

$$X[k] = \sum_{n=0}^{N-1} W_N^{kn}\ x[n]\ ,\ 0 \leq k \leq (N-1) \tag{1.3}$$

$$\boldsymbol{X}[k] = \boldsymbol{D_N}\ \boldsymbol{x}[n]\ ,\ 0 \leq (k,n) \leq (N-1) \tag{1.4}$$

Capítulo 2

Conceitos básicos

2.1 Introdução

Na primeira parte do curso, são abordados conceitos básicos relativos ao processamento digital de sinais. Inicialmente, é discutida a arquitetura de sistemas de comunicação. Em seguida, o processamento de sinais é definido, identificando-se seu objeto, seus agentes e suas ações, bem como é definida a sua arquitetura genérica. Posteriormente, é realizada uma classificação de sinais. Por fim, é apresentada uma arquitetura de sistemas de processamento digital. Nas seções que se seguem, são apresentadas diversas listagens de programas, relativas a tais assuntos.

2.2 Tipos de sinais

- O Código 2.1 define uma função para realizar a quantização de um sinal, com nível de decisão (*trigger*) no meio do intervalo de quantização.
- O Código 2.2 apresenta uma relação entre os seguintes tipos de sinais: analógico sem quantização, analógico quantizado, digital sem quantização e digital quantizado.

Código 2.1: Função para quantização em *half grid*.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: quant_half_grid.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function xq = quant_half_grid(x, min_grid, grid_step, max_grid)
%
% QUANT_HALF_GRID quantization with half grid trigger.
%
xq = x;
half_grid = grid_step / 2;
for k = 1:length(x)
    for grid_val = min_grid:grid_step:max_grid
        dist = x(k) - grid_val;
        if ( (abs(dist) < half_grid) ...
             || ...
             ( (dist < 0) && (abs(dist) == half_grid) ) ...
           )
            xq(k) = grid_val;
```

```
                break
            end
        end
end
%
% EOF
%
```

Código 2.2: Exemplos de sinais de diferentes tipos.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: tipos_de_sinais.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                          %
% Demos para                               %
% Apostila de DSP                          %
%                                          %
% Topico: tipos de sinais                  %
%                                          %
% Autor : Alexandre Santos de la Vega      %
% Modifs: 2022-1/2021-2/2011-02/2010-01/   %
%                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


% limpa ambiente
clear all
close all


% define parametros
A0   = 1;
F0   = 100;
T0   = 1/F0;
%
% numero de periodos visualizados
NOP  = 2;
%
% numero de pontos por periodo para sinal analogico
PPPa = 1000;
Tsa  = T0/PPPa;
t    = 0:Tsa:(NOP*T0);
%
% numero de pontos por periodo para sinal amostrado
PPPs = 10;
Tss  = T0/PPPs;
nTs  = 0:Tss:(NOP*T0);
%
n    = 0:(length(nTs) - 1);

```

```
% constroi sinais
xa = A0*cos(2*pi*F0*t);     % sinal analogico
xs = A0*cos(2*pi*F0*nTs);   % sinal amostrado


% quantiza sinais usando funcao definida externamente
%
% numero de intervalos do grid
NOI = 5;
% sinal analogico quantizado
xaq = quant_half_grid(xa,-A0,(A0/NOI),A0);
% sinal amostrado quantizado
xsq = quant_half_grid(xs,-A0,(A0/NOI),A0);


%
% desenha graficos
%

%
figure(1)
%
% black = 0.0 ; gray_suave = 0.8 ; white = 1.0
BCV = 1.0;
back_color = [BCV,BCV,BCV];

%
%   sem quantizacao
%
subplot(3,3,1)
plot(t,xa)
ylabel('x(t)')
xlabel('t  (s)')
%
set(gca,'Color',back_color)
title({'Sinal analógico: x(t)' , ...
       ['T_s= ' , num2str(T0/PPPs), 's' , '  ;  ' , ...
        'Q_{res}= ' , num2str(1/NOI)]})
%
subplot(3,3,4)
stem(nTs,xs)
ylabel('x(nT_s)')
xlabel('nT_s  (s)')
title({'Sinal amostrado' , 'x(nT_s) = x(t)  ;  t = nT_s'})
%
set(gca,'Color',back_color)
%
subplot(3,3,7)
stem(n,xs)
ylabel('x[n]')
xlabel('n')
%
set(gca,'Color',back_color)
title({'Sequência amostrada' , 'x[n] = x(nT_s)'})
%
%   com quantizacao
%
subplot(3,3,2)
plot(t,xa,'b', t,xaq,'r')
```

```
107 ylabel('\{x(t)\}_Q')
    xlabel('t  (s)')
109 title({'Sinal quantizado' , '\{x(t)\}_Q'})
    %
111 set(gca,'Color',back_color)
    %
113 subplot(3,3,5)
    % 'funciona, mas as cores nao ficam ok! ...'
115 %stem(nTs,[xs ; xsq]')
    hold on
117 stem(nTs,xs,'bo-')
    stem(nTs,xsq,'ro-')
119 ylabel('\{x(nT_s)\}_Q')
    xlabel('nT_s  (s)')
121 title({'Sinal digital' , ...
        '\{x(nT_s)\}_Q = \{x(t)\}_Q  ;  t = nT_s'})
123 %
    set(gca,'Color',back_color)
125 %
    subplot(3,3,8)
127 % 'funciona, mas as cores nao ficam ok! ...'
    %stem(n,[xs ; xsq]')
129 hold on
    stem(n,xs,'bo-')
131 stem(n,xsq,'ro-')
    ylabel('\{x[n]\}_Q')
133 xlabel('n')
    title({'Sequência digital' , '\{x[n]\}_Q = \{x(nT_s)\}_Q'})
135 %
    set(gca,'Color',back_color)
137 %
    %   com quantizacao e normalizado
139 %
    subplot(3,3,3)
141 plot(t,NOI*xa,'b', t,NOI*xaq,'r')
    %
143 v = axis;
    v(3) = -NOI;
145 v(4) =  NOI;
    axis(v)
147 %
    ylabel('\{x(t)\}_{Qn}')
149 xlabel('t  (s)')
    title({'Sinal quantizado normalizado' , '\{x(t)\}_{Qn}'})
151 %
    set(gca,'Color',back_color)
153 %
    subplot(3,3,6)
155 % 'funciona, mas as cores nao ficam ok! ...'
    %stem(nTs,[xs ; xsq]')
157 hold on
    stem(nTs,NOI*xs,'bo-')
159 stem(nTs,NOI*xsq,'ro-')
    %
161 v = axis;
    v(3) = -NOI;
163 v(4) =  NOI;
    axis(v)
165 %
```

```
ylabel('\{x(nT_s)\}_{Qn}')
xlabel('nT_s  (s)')
title({'Sinal digital normalizado' , ...
       '\{x(nT_s)\}_{Qn} = \{x(t)\}_{Qn}  ;  t = nT_s'})
%
set(gca,'Color',back_color)
%
subplot(3,3,9)
% 'funciona, mas as cores nao ficam ok! ...'
%stem(n,[xs ; xsq]')
hold on
stem(n,NOI*xs,'bo-')
stem(n,NOI*xsq,'ro-')
%
v = axis;
v(3) = -NOI;
v(4) =  NOI;
axis(v)
%
ylabel('\{x[n]\}_{Qn}')
xlabel('n')
title({'Sequência digital normalizada' , ...
       '\{x[n]\}_{Qn} = \{x(nT_s)\}_{Qn}'})
%
set(gca,'Color',back_color)

%%
```

2.3 Amostragem

- O Código 2.3 apresenta a geração de uma sequência discreta a partir de um sinal analógico.
- O Código 2.4 apresenta um exemplo que ilustra a influência do valor da taxa de amostragem.
- O Código 2.5 apresenta um exemplo de amostragem de *cos()*.

Código 2.3: Geração de sequência discreta a partir de sinal analógico.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: disc_seq_from_ana_sig.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       |
% Exemplo sobre                          %
% geracao de sequência discreta          %
% a partir de sinal analogico            %
%                                        %
% Autor : Alexandre S. de la Vega        %
% Versao: 2018_2                         %
%                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%
```

```
20 %
   clear all
22 close all

24 %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%

26
   %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%
28
   %
30 A0    =      1;
   f0    =   1e3;    %  f0 = 1 kHz
32 phi0 = -pi/4;

34 %
   N0      =       10;
36 Omega0 = 2*pi/N0;

38 %
   Fs = N0 * f0;
40
   %
42 NOP = 3;    %  Nbr Of Periods

44 %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%

46
   %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%
48
   %
50 NOPPP = 100;    %  Nbr Pts Per Period

52 %
   t_min  = 0;
54 t_step = (1/(f0*NOPPP));
   t_max  = (NOP/f0);
56 %
   t = t_min:t_step:t_max;
58 %
   xt = A0 * cos(2*pi*f0*t + phi0);
60
   %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%
62
   %
64 n_min  =       0;
   n_max  = (3*N0);
66 %
   n = n_min:n_max;
68 %
   xn = A0 * cos(Omega0*n + phi0);
70
   %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%
72

74 %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%

76 %
   FigNbr = 0;
```

```
78 %
   AmpSF    = 1.25;
80 EfctAmp = (AmpSF*A0);

82 %%%%%%%%%% %%%%%%%%%%

84 %
   FigNbr = FigNbr + 1;
86 hf=figure(FigNbr);
   set(hf,"papertype","a4")
88 %
   plot(t,xt,'k')
90 v(2) =    t_max;
   v(3) = -EfctAmp;
92 v(4) =  EfctAmp;
   axis(v);
94 title('Sinal analógico')
   ylabel('x(t)')
96 xlabel('t  (s)')
   %
98
   %%%%%%%%%% %%%%%%%%%%
100
   %
102 FigNbr = FigNbr + 1;
   hf=figure(FigNbr);
104 set(hf,"papertype","a4")
   %
106 plot(t,xt,'k')
   hold on
108 stem(n/Fs,xn,'r')
   %
110 v(2) =    t_max;
   v(3) = -EfctAmp;
112 v(4) =  EfctAmp;
   axis(v);
114 title('Sinal analógico e Sinal analógico amostrado')
   ylabel('x(t)   e   x(nTs)')
116 xlabel('t  (s)')
   %
118
   %%%%%%%%%% %%%%%%%%%%
120
   %
122 FigNbr = FigNbr + 1;
   hf=figure(FigNbr);
124 set(hf,"papertype","a4")
   %
126 stem(n/Fs,xn,'r')
   v(2) =    t_max;
128 v(3) = -EfctAmp;
   v(4) =  EfctAmp;
130 axis(v);
   title('Sinal analógico amostrado')
132 ylabel('x(nTs)')
   xlabel('nTs  (s)')
134 %

136 %%%%%%%%%% %%%%%%%%%%
```

```
%
FigNbr = FigNbr + 1;
hf=figure(FigNbr);
set(hf,"papertype","a4")
%
stem(n,xn,'b')
v(2) =    n_max;
v(3) = -EfctAmp;
v(4) =  EfctAmp;
axis(v);
title('Sinal discreto')
ylabel('x[n]')
xlabel('n')
%

%%%%%%%%%% %%%%%%%%%%

%
FigNbr = FigNbr + 1;
hf=figure(FigNbr);
set(hf,"papertype","a4")
%
subplot(4,1,1)
plot(t,xt,'k')
v(2) =    t_max;
v(3) = -EfctAmp;
v(4) =  EfctAmp;
axis(v);
title('Sinal analógico')
ylabel('x(t)')
xlabel('t  (s)')
%
%
subplot(4,1,2)
plot(t,xt,'k')
hold on
stem(n/Fs,xn,'r')
%
v(2) =    t_max;
v(3) = -EfctAmp;
v(4) =  EfctAmp;
axis(v);
title('Sinal analógico e Sinal analógico amostrado')
ylabel('x(t)   e   x(nTs)')
xlabel('t  (s)')
%
%
subplot(4,1,3)
stem(n/Fs,xn,'r')
v(2) =    t_max;
v(3) = -EfctAmp;
v(4) =  EfctAmp;
axis(v);
title('Sinal analógico amostrado')
ylabel('x(nTs)')
xlabel('nTs  (s)')
%
%
```

```
subplot(4,1,4)
stem(n,xn,'b')
v(2) =     n_max;
v(3) = -EfctAmp;
v(4) =  EfctAmp;
axis(v);
title('Sinal discreto')
ylabel('x[n]')
xlabel('n')
%

%
% EOF
%
```

Código 2.4: Influência do valor da taxa de amostragem.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: ana_smp_seq_3_Fs_snd_2021_10_30.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                          %
% Exemplo sobre:                                           %
%                                                          %
%    - Sinal senoidal analogico.                           %
%    - Grafico (da aproximacao) de sinal analogico.        %
%    - Amostragem de sinal continuo.                       %
%    - Sinal senoidal amostrado.                           %
%    - Grafico de sinal amostrado.                         %
%    - Geracao de som por interpolacao.                    %
%    - Importancia da taxa de amostragem original.         %
%                                                          %
% Autor : Alexandre S. de la Vega                          %
%                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

%
% ---------- ---------- ---------- ---------- ----------
%

% limpeza do ambiente de trabalho
clear all
close all

%
% ---------- ---------- ---------- ---------- ----------
%

%
% Eh suposto um sinal senoidal analogico,
%    com amplitude A, frequencia f (Hz) ou w (rad/s)
```

```
41 %   e angulo de fase Theta (rad).
   %
43 % Eh definida uma frequencia de amostragem Fs=1/Ts (Hz).
   %
45 % Eh definida uma frequencia discreta Omega=w*Ts (rad).
   %
47
   %
49 % Definicoes gerais
   %
51
   % parametros do sinal analogico
53 A = 1;
   %
55 f = 1e3;          % em Hertz = 1/segundo.
   w = 2*pi*f;       % em rad/s.
57 %
   Theta = 0;        % em rad.
59
   % frequencia de amostragem
61 Fs = 44.1e3;     % em Hertz = 1/segundo.

63 % parametros do sinal amostrado
   Omega = w/Fs;   % Omega = w*Ts   (rad)
65
   %
67 % ---------- ---------- ---------- ---------- ----------
   %
69
   %
71 % Eh definido um tempo total de gravacao Trec.
   %
73 % Eh calculada a quantidade de amostras Nrec.
   %
75 % Sao calculadas as amostras x_rec.
   %
77
   %
79 Trec = 2;   % em segundos.
   %
81 %  Nrec = fix(Trec/Ts) + 1
   %       = fix(Trec*Fs) + 1
83
   %
85 Nrec = fix(Trec*Fs) + 1;

87 %
   n_rec = 0:(Nrec-1);
89 %
   x_rec = A * cos( (Omega*n_rec) + Theta);
91
   %
93 % ---------- ---------- ---------- ---------- ----------
   %
95
   %
97 % Eh definido o tempo de 1 periodo fundamental Tf=1/f.
   %
99 % Eh calculada a quantidade de amostras Nf.
```

```
%
% Sao separadas as amostras x_f a partir das amostras x_rec.
%

%
%  Nf = fix(Tcos/Ts) + 1
%     = fix(Fs/fcos) + 1
%
Nf = fix(Fs/f) + 1;

%
n_f = 0:(Nf-1);
%
%%%x_f = A * cos( (Omega*n_f) + Theta);
%
x_f = x_rec(1:Nf);

%
% ---------- ---------- ---------- ---------- ----------
%

%
% Sao realizados diversos graficos
%

%
% Definicoes gerais
%

% inicializa o numero do canvas
FigNbr = 0;

% calcula faixa de tempo de 1 periodo fundamental
t_f = (n_f / Fs);  % t = n*Ts  (s)

% calcula faixa de tempo de gravacao
%t_rec = (n_rec / Fs);  % t = n*Ts  (s)

%
% ---------- ---------- ----------
%

%
% Eh desenhado 1 periodo fundamental
%   (da aproximacao) do sinal analogico.
%

%
teo_str = 'Sinal senoidal analógico:  período fundamental.';
%
signal_str = 'x(t) = A cos( (2 \pi f t) + \Theta)';
%
sep_str =  '  ;  ';
%
param_str = [                    'A = '     , num2str(A)
];     |
param_str = [param_str, sep_str, 'f = '     , num2str(f)     , ' (Hz)' ];
param_str = [param_str, sep_str, '\Theta = ', num2str(Theta), ' (rad)'];
%
```

```
ana_tit_str = {teo_str, signal_str, param_str};

%
FigNbr = FigNbr + 1;
figure(FigNbr)
%
plot(t_f, x_f, "k")
%
v    = axis;
v(2) = t_f(end);
axis(v)
%
title(ana_tit_str)
ylabel('x (t)')
xlabel('t  (s)')

% para teste...
%return

%
% ---------- ---------- ----------
%

%
% Eh desenhado 1 periodo fundamental
%   do sinal amostrado com Fs.
%

%
teo_str = 'Sinal senoidal amostrado:  período fundamental.';
%
signal_str = 'x (nTs)  =  A cos ( (2 \pi f n Ts) + \Theta)';
%
sep_str =  '  ;  ';
%
param_str = [                    'A = '        , num2str(A)
];      |
param_str = [param_str, sep_str, 'f = '        , num2str(f)     , ' (Hz)' ];
param_str = [param_str, sep_str, '\Theta = '   , num2str(Theta), ' (rad)'];
param_str = [param_str, sep_str, 'Ts = '       , num2str(1/Fs) , ' (s)'
];      |
param_str = [param_str, sep_str, 'Fs = 1/Ts = ', num2str(Fs)    , ' (Hz)' ];
%
smp_tit_str = {teo_str, signal_str, param_str};

%
FigNbr = FigNbr + 1;
figure(FigNbr)
%
stem(t_f, x_f, "k")
%
v    = axis;
v(2) = t_f(end);
axis(v)
%
title(smp_tit_str)
ylabel('x (n Ts)')
xlabel('n Ts  (s)')
```

```
215 % para teste...
    %return
217
    %
219 % ---------- ---------- ----------
    %
221
    %
223 % Eh desenhado 1 periodo fundamental
    %   da sequencia amostrada com Fs.
225 %

227 %
    teo_str = 'Sequência senoidal:  período fundamental.';
229 %
    signal_str = 'x [n]  =  A cos ( ( (2 \pi f Ts) n) + \Theta)';
231 signal_str = [signal_str, '  =  A cos ( (\Omega n) + \Theta)'];
    %
233 map_str = '\Omega  =  \omega Ts  =  (2 \pi f) Ts';
    %
235 sep_str =  '  ;  ';
    %
237 param_str = [                    'A = '        , num2str(A)
    ];      |
    param_str = [param_str, sep_str, 'f = '        , num2str(f)     , ' (Hz)' ];
239 param_str = [param_str, sep_str, '\Theta = '   , num2str(Theta), ' (rad)'];
    param_str = [param_str, sep_str, 'Ts = '       , num2str(1/Fs) , ' (s)'
    ];      |
241 param_str = [param_str, sep_str, 'Fs = 1/Ts = ', num2str(Fs)    , ' (Hz)' ];
    %
243 seq_tit_str = {teo_str, signal_str, map_str, param_str};

245 %
    FigNbr = FigNbr + 1;
247 figure(FigNbr)
    %
249 stem(n_f, x_f, "k")
    %
251 v    = axis;
    v(2) = n_f(end);
253 axis(v)
    %
255 title(seq_tit_str)
    ylabel('x [n]')
257 xlabel('n')

259 % para teste...
    %return
261
    %
263 % ---------- ---------- ----------
    %
265
    %
267 % Eh desenhado 1 periodo fundamental
    %   conjunto:
269 %     (da aproximacao) do sinal analogico,
    %     do sinal amostrado com Fs
271 %     e
```

```
%      da sequencia amostrada com Fs.
%

%
FigNbr = FigNbr + 1;
figure(FigNbr)
%
%
subplot(3,1,1)
%
plot(t_f, x_f, "k")
%
v    = axis;
v(2) = t_f(end);
axis(v)
%
title({'Geração de x(n Ts) por amostragem de x(t).',
       'Geração de x[n] por indexação de x(n Ts).'})
%
ylabel('x (t)')
xlabel('t  (s)')
%
%
subplot(3,1,2)
%
stem(t_f, x_f, "k")
%
v    = axis;
v(2) = t_f(end);
axis(v)
%
ylabel('x (n Ts)')
xlabel('n Ts  (s)')
%
%
subplot(3,1,3)
%
stem(n_f, x_f, "k")
%
v    = axis;
v(2) = n_f(end);
axis(v)
%
ylabel('x [n]')
xlabel('n')

% para teste...
%return

%
% ---------- ---------- ----------
%

%
% Eh desenhado 1 periodo fundamental
%   conjunto:
%      da recuperacao
%      do sinal amostrado
%      com Fs diferentes.
```

```
331 %

333 %
    FigNbr = FigNbr + 1;
335 figure(FigNbr)
    %
337 %
    subplot(3,1,1)
339 %
    stem(t_f, x_f, "k")
341 %
    v    = axis;
343 v(2) = 2*t_f(end);
    axis(v)
345 %
    title('Geração de x(n Ts) a partir de x[n], empregando Fs = 1/Ts.')
347 ylabel('x (n Ts)')
    %
349 %
    subplot(3,1,2)
351 %
    stem((2*t_f), x_f, "k")
353 %
    v    = axis;
355 v(2) = 2*t_f(end);
    axis(v)
357 %
    title('Geração de x(n Ts) a partir de x[n], empregando (Fs/2) = 1/(2 Ts).')
359 ylabel('x (n Ts)')
    %
361 %
    subplot(3,1,3)
363 %
    stem((0.5*t_f), x_f, "k")
365 %
    v    = axis;
367 v(2) = 2*t_f(end);
    axis(v)
369 %
    title('Geração de x(n Ts) a partir de x[n], empregando (2 Fs) = 1/(Ts/2).')
371 ylabel('x (n Ts)')
    %
373 xlabel('n Ts  (s)')

375 % para teste...
    %return
377
    %
379 % ---------- ---------- ----------
    %
381
    %
383 % Eh desenhado 1 periodo fundamental
    %   conjunto:
385 %     da recuperacao
    %     do sinal analogico
387 %     com Fs diferentes.
    %
389
```

```
%
FigNbr = FigNbr + 1;
figure(FigNbr)
%
%
subplot(3,1,1)
%
plot(t_f, x_f, "k")
%
v    = axis;
v(2) = 2*t_f(end);
axis(v)
%
title('Geração de x(t) a partir de x[n], empregando Fs = 1/Ts.')
ylabel('x (t)')
%
%
subplot(3,1,2)
%
plot((2*t_f), x_f, "k")
%
v    = axis;
v(2) = 2*t_f(end);
axis(v)
%
title('Geração de x(t) a partir de x[n], empregando (Fs/2) = 1/(2 Ts).')
ylabel('x (t)')
%
%
subplot(3,1,3)
%
plot((0.5*t_f), x_f, "k")
%
v    = axis;
v(2) = 2*t_f(end);
axis(v)
%
title('Geração de x(t) a partir de x[n], empregando (2 Fs) = 1/(Ts/2).')
ylabel('x (t)')
%
xlabel('t  (s)')

% para teste...
%return

%
% ---------- ---------- ---------- ---------- ----------
%

%
% Eh considerada a frequencia de amostragem Fs original.
%
% Os valores das amostras sao enviados para a placa de som.
%
% A placa de som interpola os pontos com Fs e gera o som.
%
%
% Sao gerados sons com Fs diferentes da Fs original.
%
```

```

%
display("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n")
display("\==================================================")
display("\===  Geração de som a partir de amostras  ===")
display("\==================================================")
%
escolha = toupper( input("\nDeseja gerar o som (s/n) ?  [s] :   ", "s") );
if (escolha == "N")
  display("\n\n\n")
  return
endif
%
display("\n\n\n")
display("\n\nSom para Fs original.")
soundsc(x_rec,Fs)
%
display("\nPressione qualquer tecla para continuar...")
pause
%
display("\n\n\nSom para (Fs/2) = 1/(2 Ts).")
soundsc(x_rec,(Fs/2))
%
display("\nPressione qualquer tecla para continuar...")
pause
%
display("\n\n\nSom para (2 Fs) = 1/(Ts/2).")
soundsc(x_rec,(2*Fs))
%
display("\n\n\n")

%
% ----------- ----------- ----------- ----------- -----------
%

%
% EOF
%
```

Código 2.5: Exemplo de amostragem de *cos()*.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: cos_amostragem.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DSP - Aula 04/01/2016
%
% Autor: Alexandre Santos de la Vega
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

%
```

```
%%%%% %%%%% %%%%% %%%%% %%%%%
%
% elimina as variveis de usuario, existentes no ambiente.
clear all

% elimina os graficos criados pelo usuario, existentes no ambiente.
close all

%
%%%%% %%%%% %%%%% %%%%% %%%%%
%

% definicao de parametros gerais
%F0 = 1e3;
%F0 = 4.4e3;
%F0 = 10e3;
F0 = 17.6e3;
%
Fs = 44e3;
%
T0 = 1/F0;
Ts = 1/Fs;
%
Trec = 12*T0;
%
FigNbr = 0;

%
%%%%% %%%%% %%%%% %%%%% %%%%%
%

% simulacao de sinal analogico
%
%   definicao de parametros
%
% Obs.: NPP = V eh equivalente a Fs = V*F0 ...
%
NumPtsPer = 200;
Tstep = T0 / NumPtsPer;
nT = 0:Tstep:(Trec-Tstep);
%
%   definicao da funcao
xa = cos(2*pi*F0*nT);

%   grafico
FigNbr = FigNbr + 1;
figure(FigNbr)
%
plot(nT,xa,'r')
%
V = axis;
axis([V(1) Tstep*length(xa) V(3) V(4)])
%
title('Simulação de curva analógica')
ylabel('x_a (t)')
xlabel('t (s)')

%
```

```
76 %%%%% %%%%% %%%%% %%%%% %%%%%
   %
78
   % simulacao de amostragem
80 %
   %   definicao de parametros
82 %
   n = 0:fix(Trec/Ts);
84 %
   %   definicao da funcao
86 xd = cos(2*pi*F0*Ts*n);

88 %   grafico do sinal amostrado
   FigNbr = FigNbr + 1;
90 figure(FigNbr)
   %
92 plot(nT,xa,'r')
   hold on
94 stem(n*Ts,xd,'k')
   %
96 V = axis;
   axis([V(1) Tstep*length(xa) V(3) V(4)])
98 %
   title('Simulação de amostragem: sinal original e sinal amostrado')
100 ylabel('x_a (t)   e   x_a (n Ts)')
   xlabel('t (s)')
102 hold off

104 %   grafico do sinal discreto
   FigNbr = FigNbr + 1;
106 figure(FigNbr)
   %
108 stem(n,xd,'k')
   %
110 V = axis;
   axis([V(1) length(xd)-1 V(3) V(4)])
112 %
   title('Simulação de amostragem: sinal discreto')
114 ylabel('x [n]')
   xlabel('n')
116
   %
118 %%%%% %%%%% %%%%% %%%%% %%%%%
   %
120
   % reuniao de graficos
122 %
   FigNbr = FigNbr + 1;
124 figure(FigNbr)
   %
126 %
   subplot(3,1,1)
128 plot(nT,xa,'r')
   %
130 V = axis;
   axis([V(1) Tstep*length(xa) V(3) V(4)])
132 %
   title('Simulação de curva analógica')
134 ylabel('x_a (t)')
```

```
xlabel('t (s)')
%
%
subplot(3,1,2)
plot(nT,xa,'r')
hold on
stem(n*Ts,xd,'k')
%
V = axis;
axis([V(1) Tstep*length(xa) V(3) V(4)])
%
title('Simulação de amostragem: sinal original e sinal amostrado')
ylabel('x_a (t)   e   x_a (n Ts)')
xlabel('t (s)')
hold off
%
%
subplot(3,1,3)
stem(n,xd,'k')
%
V = axis;
axis([V(1) length(xd)-1 V(3) V(4)])
%
title('Simulação de amostragem: sinal discreto')
ylabel('x [n]')
xlabel('n')

%
% EOF
%
```

2.4 Arquitetura de sistemas de processamento digital

- O Código 2.6 apresenta os sinais envolvidos em uma cadeia de processamento digital para sinais analógicos.

Código 2.6: Sinais em cadeia de processamento digital para sinais analógicos.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: cadeia_proc_dig_sinal_ana.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Demos para                                                                  %
% Apostila de DSP                                                             %
%                                                                             %
% Topico: cadeia                                                              %
%         de processamento digital                                            %
%         de sinal analogico                                                  %
%                                                                             %
% Autor : Alexandre Santos de la Vega                                         %
% Modifs: /2010-1/2011-2/2022-1/                                              %
```

```
20 %                                                  %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
22 %


24
   % limpa ambiente
26 clear all
   close all
28


30 %
   % investiga interfaces graficas existentes
32 %available_graphics_toolkits()
   %
34 % investiga interface grafica default
   %graphics_toolkit()
36 %
   % experimenta diferentes interfaces graficas
38 %graphics_toolkit("fltk")
   %graphics_toolkit("gnuplot")
40 %graphics_toolkit("qt")
   %
42


44 % define parametros
   A0  = 1;
46 F0  = 1e3;
   T0  = 1/F0;
48 %
   NOP = 2;      % numero de periodos visualizados
50 %
   PPPa = 1000;  % numero de pontos por periodo para sinal analogico
52 Tsa  = T0/PPPa;
   t    = 0:Tsa:(NOP*T0);
54 %
   PPPs = 10;    % numero de pontos por periodo para sinal amostrado
56 Tss  = T0/PPPs;
   nTs  = 0:Tss:(NOP*T0);
58 %
   n    = 0:(length(nTs) - 1);
60


62 % constroi sinais
   xa = A0*cos(2*pi*F0*t);     % sinal analogico
64 xs = A0*cos(2*pi*F0*nTs);   % sinal amostrado


66
   % quantiza sinais usando funcao definida externamente
68 NOI = 5;  % numero de intervalos do grid
   xaq = quant_half_grid(xa,-A0,(A0/NOI),A0);  % sinal analogico quantizado
70 xsq = quant_half_grid(xs,-A0,(A0/NOI),A0);  % sinal amostrado quantizado


72
   % desenha graficos
74 %
   back_val = 1.0;
76
   figure(1)
78 %
```

```
subplot(3,2,1)
plot(t,xa,'k')
set(gca,"color",[back_val,back_val,back_val])
ylabel('x(t)')
xlabel('t (s)')
title({'Sinal de entrada analógico', 'ou', ...
       'Sinal na saída do filtro anti-aliasing'})
%
subplot(3,2,3)
plot(t,xa,'k')
hold on
for k = 1:(length(xs)-1)
%    stem(nTs(k),xs(k),':b')  % nao ficou bom...
    plot([nTs(k)   nTs(k+1)] , [xs(k) xs(k)  ] , 'b')
    plot([nTs(k+1) nTs(k+1)] , [xs(k) xs(k+1)] , 'b')
end
set(gca,"color",[back_val,back_val,back_val])
ylabel('r(t)')
xlabel('t (s)')
title('Sinal na saída do S/H')
%
subplot(3,2,5)
plot(t,xa,'k')
hold on
for k = 1:(length(xs)-1)
    plot([nTs(k)   nTs(k+1)] , [xs(k) xs(k)  ] , 'b')
    plot([nTs(k+1) nTs(k+1)] , [xs(k) xs(k+1)] , 'b')
end
for k = 1:(length(xsq)-1)
    plot([nTs(k)   nTs(k+1)] , [xsq(k) xsq(k)  ] , 'r')
    plot([nTs(k+1) nTs(k+1)] , [xsq(k) xsq(k+1)] , 'r')
end
set(gca,"color",[back_val,back_val,back_val])
ylabel('c(t)')
xlabel('t (s)')
title('Sinal na saída do ADC')
%
subplot(3,2,2)
plot((t*F0*PPPs),xa,'k')
hold on
stem(n,xs,'k')
stem(n,xsq,'r')
set(gca,"color",[back_val,back_val,back_val])
ylabel('[ x[n] ]_Q')
xlabel('n')
title({'Sequência digital de entrada', '=', ...
       'Sequência digital de saída'})
%
subplot(3,2,4)
stem(n,xsq,':r')
hold on
for k = 1:(length(xsq)-1)
    plot([n(k)   n(k+1)] , [xsq(k) xsq(k)  ] , 'r')
    plot([n(k+1) n(k+1)] , [xsq(k) xsq(k+1)] , 'r')
end
set(gca,"color",[back_val,back_val,back_val])
ylabel('c(t)')
xlabel('t (s)')
title('Sinal na saída do DAC')
```

```
138 %
    subplot(3,2,6)
140 plot(t,xa,'k')
    hold on
142 plot(nTs,xsq,'r')
    set(gca,"color",[back_val,back_val,back_val])
144 ylabel('x(t)')
    xlabel('t (s)')
146 title({'Sinal na saída do filtro de smoothing', '=', ...
        'Sinal de saída analógico'})
148
    %
150 % EOF
    %
```

Parte II

Sinais e sistemas no domínio do tempo

Capítulo 3

Sinais no domínio do tempo

3.1 Introdução

Na segunda parte do curso, são abordados os seguintes itens sobre sinais e sistemas (com tempo discreto) no domínio do tempo: definições, classificações, exemplos e caracterizações de sinais e sistemas. Além disso, são também abordados os seguintes tópicos sobre Sistema Linear e Invariante ao Tempo (SLIT) com tempo discreto: características, representações e cálculo de resposta. Nas seções que se seguem, são apresentadas diversas listagens de programas, relativas a tais assuntos.

3.2 Tipos de sequências

3.2.1 Sistema numérico

- O Código 3.1 apresenta um exemplo contendo dois sinais reais e um sinal complexo formado por tais sinais reais.

Código 3.1: Tipos de sequências - sistema numérico.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %
3  % Arquivo: tipos_de_seqs_sist_num.m
   %
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

7
   %
9  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %                                                                             %
11 % Demos para                                                                  %
   % Apostila de DSP                                                            %
13 %                                                                             %
   % Topico: tipos de sequencias                                                %
15 %             sistema numerico                                                %
   %                                                                             %
17 % Autor : Alexandre Santos de la Vega                                         %
   % Modifs: /2012-02/                                                          %
19 %                                                                             %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
21 %
```

```
23
% limpa ambiente
25 clear all
close all
27

29 % constroi sinais
vr =      [ 1  2  3  4  5  6  5  4  3  2  1];  % sinal real
31 wr = 4 * [ 1  1  1  0  0  0  0  0 -1 -1 -1];  % sinal real
%
33 xc = vr + j * wr;  % sinal complexo

35
% define indice
37 n = 0:(length(vr)-1);

39
% desenha graficos
41 %
%   lembrete: AXIS([XMIN XMAX YMIN YMAX]);
43 %
figure(1)
45 %
%   sequencias reais
47 %
subplot(2,2,1)
49 stem(n,vr,'k')
ylabel('v_r [n]')
51 title('Sequências reais: v_r [n]  e  w_r [n]')
v = axis;
53 AXIS([v(1) v(2) -6 6]);
%
55 subplot(2,2,3)
stem(n,wr,'k')
57 ylabel('w_r [n]')
xlabel('n')
59 v = axis;
AXIS([v(1) v(2) -6 6]);
61 %
%   sequencia complexa
63 %
subplot(2,2,2)
65 stem(n,abs(xc),'b')
ylabel('| x_c [n] |')
67 title('Sequência complexa: x_c [n] = v_r [n] + j \cdot w_r [n]')
v = axis;
69 AXIS([v(1) v(2) -6 6]);
%
71 subplot(2,2,4)
stem(n,angle(xc),'r')
73 ylabel('\angle x_c [n]    (rad)')
xlabel('n')
75 v = axis;
AXIS([v(1) v(2) -pi pi]);
77

79 %
% EOF
81 %
```

3.2.2 Comprimento

- O Código 3.2 apresenta exemplos de sequências com comprimento finito e infinito.

Código 3.2: Tipos de sequências - comprimento.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: tipos_de_seqs_comprimento.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Demos para                                                                  %
% Apostila de DSP                                                             %
%                                                                             %
% Topico: tipos de sequencias                                                 %
%           comprimento                                                       %
%                                                                             %
% Autor : Alexandre Santos de la Vega                                         %
% Modifs: /2012-02/                                                           %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


% limpa ambiente
clear all
close all


% constroi sinais
%
xfin = [ 0  0  0  1  2  3  0  0  0  3  2  1  0  0  0];
%
xz = zeros(1,41);
%
xld1 = xz;
xld1(11:end) = 1;
%
xld2 = xz;
xld2(21:end) = 1;
%
xld3 = xz;
xld3(31:end) = 1;
%
xle1 = fliplr(xld1);
xle2 = fliplr(xld2);
xle3 = fliplr(xld3);


% define indices
%
nf1 = -20:-6;
nf2 =  -7:7;
nf3 =   6:20;
```

```
%
ni = -20:20;


% desenha graficos
%
%   lembrete: AXIS([XMIN XMAX YMIN YMAX]);
%
figure(1)
%
%   sequencias de comprimento finito
%
subplot(3,1,1)
stem(nf1,xfin,'k')
title({'Sequências de comprimento finito -- N = 15', ...
       'x_1 [n] = x_2 [n+13] = x_3 [n+26]'})
ylabel('x_1 [n]')
v = axis;
axis([-20 20 v(3) v(4)]);
%
subplot(3,1,2)
stem(nf2,xfin,'k')
title('x_2 [n] = x_1 [n-13] = x_3 [n+13]')
ylabel('x_2 [n]')
v = axis;
axis([-20 20 v(3) v(4)]);
%
subplot(3,1,3)
stem(nf3,xfin,'k')
title('x_3 [n] = x_1 [n-26] = x_2 [n-13]')
ylabel('x_3 [n]')
xlabel('n')
v = axis;
axis([-20 20 v(3) v(4)]);
%
%
figure(2)
%
%   sequencias de comprimento infinito
%
%
%   lateral esquerda
%
subplot(3,2,1)
stem(ni,xle1,'k')
title({'Sequências de comprimento infinito', ...
       'Lateral esquerda'})
ylabel('u [-n+10]')
%
subplot(3,2,3)
stem(ni,xle2,'k')
title('Lateral esquerda - Anticausal')
ylabel('u [-n]')
%
subplot(3,2,5)
stem(ni,xle3,'k')
title('Lateral esquerda')
ylabel('u [-n-10]')
xlabel('n')
```

```
%
%   lateral direita
%
subplot(3,2,2)
stem(ni,xld1,'k')
title({'Sequências de comprimento infinito', ...
       'Lateral direita'})
ylabel('u [n+10]')
%
subplot(3,2,4)
stem(ni,xld2,'k')
title('Lateral direita - Causal')
ylabel('u [n]')
%
subplot(3,2,6)
stem(ni,xld3,'k')
title('Lateral direita')
ylabel('u [n-10]')
xlabel('n')


%
% EOF
%
```

3.2.3 Simetria

- O Código 3.3 apresenta exemplos de simetria em sinais reais e complexos.

Código 3.3: Tipos de sequências - simetria.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: tipos_de_seqs_simetria.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
% Demos para                                                                   %
% Apostila de DSP                                                              %
%                                                                              %
% Topico: simetrias                                                            %
%                                                                              %
% Autor : Alexandre Santos de la Vega                                          %
% Modifs: /2010-01/2011-02/                                                    %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


% limpa ambiente
clear all
close all


```

```
28 % define parametros
   N = 50;
30 n = round(-(N/2)):round((N/2));

32
   % constroi sinais
34 xr    = rand(1,N+1);           % sequencia real
   %
36 x_re = rand(1,N+1);
   x_im = rand(1,N+1);
38 xc    = (x_re + j * x_im);  % sequencia complexa

40
   % calcula sequencias par e impar
42 x_e = 0.5 * (xr + fliplr(xr));
   x_o = 0.5 * (xr - fliplr(xr));
44

46 % calcula sequencias conjugada simetrica e conjugada anti-simetrica
   xc_cs = 0.5 * (xc + conj(fliplr(xc)));
48 xc_ca = 0.5 * (xc - conj(fliplr(xc)));

50
   % desenha graficos
52 %
   figure(1)
54 %
   %   sequencia real
56 %
   subplot(3,1,1)
58 stem(n,xr)
   ylabel('x[n]')
60 xlabel('n')
   title('Sequência real: x[n] = x_e[n] + x_o[n]')
62 %
   subplot(3,1,2)
64 stem(n,x_e)
   ylabel('x_e[n]')
66 xlabel('n')
   title('Sequência par: x_e[n]')
68 %
   subplot(3,1,3)
70 stem(n,x_o)
   ylabel('x_o[n]')
72 xlabel('n')
   title('Sequência ímpar: x_o[n]')
74 %
   %
76 figure(2)
   %
78 %   sequencia complexa
   %
80 %     modulo
   %
82 subplot(3,2,1)
   stem(n,real(xc))
84 ylabel('Re \{ x[n] \}')
   xlabel('n')
86 title('Parte real da sequência complexa:
```

```
        Re \{ x[n] \} = Re \{ x_{cs}[n] +  x_{ca}[n] \} ')
%
subplot(3,2,3)
stem(n,real(xc_cs))
ylabel('Re \{ x_{cs}[n] \}')
xlabel('n')
title('Parte real  da sequência conjugada simétrica:
        Re \{ x_{cs}[n] \}')
%
subplot(3,2,5)
stem(n,real(xc_ca))
ylabel('Re \{ x_{ca}[n] \}')
xlabel('n')
title('Parte real  da sequência conjugada anti-simétrica:
        Re \{ x_{ca}[n] \}')
%
%     angulo de fase em graus
%
subplot(3,2,2)
stem(n,imag(xc))
ylabel('Im \{ x[n] \}')
xlabel('n')
title('Parte imaginária da sequência complexa:
        Im \{ x[n] \} = Im \{ x_{cs}[n] +  x_{ca}[n] \} ')
%
subplot(3,2,4)
stem(n,imag(xc_cs))
ylabel('Im \{ x_{cs}[n] \}')
xlabel('n')
title('Parte imaginária da sequência conjugada simétrica:
        Im \{ x_{cs}[n] \} ')
%
subplot(3,2,6)
stem(n,imag(xc_ca))
ylabel('Im \{ x_{ca}[n] \}')
xlabel('n')
title('Parte imaginária da sequência conjugada anti-simétrica:
        Im \{ x_{ca}[n] \} ')


%
% EOF
%
```

3.3 Operações básicas sobre sequências

- O Código 3.4 apresenta a operação de deslocamento circular.
- O Código 3.5 define a função "tenda".
- O Código 3.6 apresenta as operações de deslocamento e escalamento, utilizando a "tenda" analógica.
- O Código 3.7 apresenta as operações de deslocamento e escalamento, utilizando a "tenda" discreta.
- O Código 3.8 apresenta um exemplo que emprega deslocamento linear e espelhamento circular.
- O Código 3.9 e o Código 3.10 definem as sequências finitas que serão utilizadas no Código 3.13.
- O Código 3.11 e o Código 3.12 definem as sequências periódicas que serão utilizadas no Código 3.13.
- O Código 3.13 apresenta um exemplo que relaciona as convoluções linear, periódica e circular.

Código 3.4: Deslocamento circular.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %
3  % Arquivo: deslocamento_circular.m
   %
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

7
   %
9  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %                                                                       %
11 % Demos para                                                            %
   % Apostila de DSP                                                      %
13 %                                                                       %
   % Topico: deslocamento circular                                        %
15 %                                                                       %
   % Autor : Alexandre Santos de la Vega                                  %
17 % Modifs: /2010-01/2011-02/                                            %
   %                                                                       %
19 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %
21

23 % limpa ambiente
   clear all
25 close all

27 N = 10;
   n = 0:(N-1);
29
   NDl =  3;
31 NDr = -3;
```

```
33 x = n;

35 kl = mod((n+NDl),N);
   yl = x(kl+1);
37
   kr = mod((n+NDr),N);
39 yr = x(kr+1);

41 figure(1)
   %
43 subplot(2,1,1)
   stem(n,x,'b')
45 hold on
   stem(n,yl,'r')
47 ylabel('y[n] = x[<n+N_D>_N]')
   xlabel('n')
49 title('Deslocamento circular: esquerda, com N_D =  3')
   legend('x[n]', 'y[n]')
51 %
   subplot(2,1,2)
53 stem(n,x,'b')
   hold on
55 stem(n,yr,'r')
   ylabel('y[n] = x[<n+N_D>_N]')
57 xlabel('n')
   title('Deslocamento circular: direita, com N_D = -3')
59 legend('x[n]', 'y[n]')

61 %
   % EOF
63 %
```

Código 3.5: Função tenda.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %
3  % Arquivo: tenda.m
   %
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %
7  function y = tenda(t)
   %
9  % TENDA Gera a funcao exemplo tenda.
   %
11 y1 = (    2*t)+(-4);
   y2 = ( 0.25*t)+(1.25);
13 y3 = (-0.25*t)+(4.75);
   y4 = (   -2*t)+(24);
15
   y = y1.*(2<=t & t<3)  + y2.*(3<=t & t<7) + ...
17      y3.*(7<=t & t<11) + y4.*(11<=t & t<=12) ;
   %
19 % EOF
   %
```

Código 3.6: Função tenda analógica (deslocamento e escalamento).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: tenda_ana_desl_escal_t.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                   %
% Titulo: Demo de deslocamentos e   %
%         escalamentos em t         %
%                                   %
% Autor : Alexandre Santos de la Vega %
% Datas : /2009-02/2011-01/         %
%                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


% limpeza do ambiente de trabalho
%  variaveis
clear all
%  janelas
close all


% definicao dos parametros
Tmin =  -5;
Tmax =  25;
Step = 0.1;

t = Tmin:Step:Tmax;


% definicao das funcoes
%   nota: funcao tenda definida em arquivo tenda.m
Forg   = tenda(t);
Fleft  = tenda(t+5);
Fright = tenda(t-5);
Fcomp  = tenda(2*t);
Fxpand = tenda(t/2);
Frx    = tenda(2*(t-5));
Fxr    = tenda((2*t)-5);


% graficos
%
figure(1)
%
subplot(3,2,1)
plot(t,Forg)
ylabel('f(t)')
title('Deslocamentos em t')
%
subplot(3,2,3)
plot(t,Fleft)
```

```
ylabel('f_a(t) = f(t+T_D)')
title('Avanço: T_D = 5 s')
%
subplot(3,2,5)
plot(t,Fright)
ylabel('f_d(t) = f(t-T_D)')
title('Atraso: T_D = 5 s')
xlabel('t (s)')
%
subplot(3,2,2)
plot(t,Forg)
ylabel('f(t)')
title('Escalamentos em t')
%
subplot(3,2,4)
plot(t,Fcomp)
ylabel('f_c(t) = f(t*K)')
title('Compressão: K = 2')
%
subplot(3,2,6)
plot(t,Fxpand)
ylabel('f_e(t) = f(t/K)')
title('Expansão: K = 2')
xlabel('t (s)')
%
figure(2)
%
subplot(3,1,1)
plot(t,Forg)
ylabel('f(t)')
title('Deslocamentos e escalamentos em t')
%
subplot(3,1,2)
plot(t,Frx)
ylabel('f_{dc}(t) = f(2*(t-T_D))')
title('Atraso seguido de compressão: K = 2 e T_D = 5 s')
%
subplot(3,1,3)
plot(t,Fxr)
ylabel('f_{cd}(t) = f((2*t)-T_D)')
title('Compressão seguida de atraso: K = 2 e T_D = 5 s')
xlabel('t (s)')

%
% EOF
%
```

Código 3.7: Função tenda digital (deslocamento e escalamento).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: tenda_dig_desl_escal_t.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                     %
% Titulo: Demo de deslocamentos e     %
%         escalamentos em n           %
%                                     %
% Autor : Alexandre Santos de la Vega %
% Datas : /2009-02/2011-01/           %
%                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


% limpeza do ambiente de trabalho
% variaveis
clear all
% janelas
close all


% definicao dos parametros
Tmin =  -5;
Tmax =  25;
Step = 0.1;
%
t = Tmin:Step:Tmax;
%
%   conversao de tempo para indice
n = 0:(length(t)-1);
n = n + (Tmin/Step);


% definicao das funcoes
%   nota: funcao tenda definida em arquivo tenda.m
Forg   = tenda(t);
Fleft  = tenda(t+5);
Fright = tenda(t-5);
Fcomp  = tenda(2*t);
Fxpand = tenda(t/2);  % 'isso representa up-sampling + interpolacao !!!'
Frx    = tenda(2*(t-5));
Fxr    = tenda((2*t)-5);
%
%   retirando a interpolacao...
Finterp = Fxpand;
Fxpand  = zeros(1,length(t));
odd_ind = 1:2:length(t);
Fxpand(odd_ind) = Finterp(odd_ind);


% graficos
%
figure(1)
%
subplot(3,2,1)
stem(n,Forg)
ylabel('f[n]')
title('Deslocamentos em n')
%
subplot(3,2,3)
stem(n,Fleft)
```

```
    ylabel('f_a[n] = f[n+N_D]')
69  title('Avanço: N_D = 50')
    %
71  subplot(3,2,5)
    stem(n,Fright)
73  ylabel('f_d[n] = f[n-N_D]')
    title('Atraso: N_D = 50')
75  xlabel('n')
    %
77  subplot(3,2,2)
    stem(n,Forg)
79  ylabel('f[n]')
    title('Escalamentos em n')
81  %
    subplot(3,2,4)
83  stem(n,Fcomp)
    ylabel('f_{ds}[n] = f[n*K]  ')
85  title('Down-sampling: K = 2')
    %
87  subplot(3,2,6)
    stem(n,Fxpand)
89  ylabel('f_{us}[n] = f[n/K]  ')
    title('Up-sampling: K = 2')
91  xlabel('n')
    %
93  figure(2)
    %
95  subplot(3,1,1)
    stem(n,Forg)
97  ylabel('f[n]')
    title('Deslocamentos e escalamentos em n')
99  %
    subplot(3,1,2)
101 stem(n,Frx)
    ylabel('f_{dds}[n] = f[K*(n-N_D)]  ')
103 title('Atraso seguido de down-sampling: K = 2 e N_D = 50')
    %
105 subplot(3,1,3)
    stem(n,Fxr)
107 ylabel('f_{dsd}[n] = f[(K*n)-N_D]  ')
    title('Down-sampling seguido de atraso: K = 2 e N_D = 50')
109 xlabel('n')

111 %
    % EOF
113 %
```

Código 3.8: Deslocamento linear e espelhamento circular.

```
1   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
3   % Arquivo: desloc_linear_espelh_circ.m
    %
5   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

7

    %
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                      %
% Titulo: Demo de deslocamento linear  %
%         e                            %
%         espelhamento circular        %
%                                      %
% Autor : Alexandre Santos de la Vega  %
% Datas : /2012-01/                    %
%                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% limpeza do ambiente de trabalho
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%  variaveis
clear all
%  janelas
close all


%%%%%%%%%%%%
% calculos
%%%%%%%%%%%%

%
x = [  2;  1;  2; -2; ...
      -1; -2;  1;  2; ...
       1; -1; -2; -1; ...
       2;  0; -2; -1; ...
       0;  1; -2;  0; ...
       2;  1;  0; -1];

%
N = 4;
Lx = length(x);

%
xm = reshape(x, N, (Lx/N));

%
ym = [xm; xm(1,:)];
ym = flipud(ym);
ym = ym(1:(end-1),:);

%
y = reshape(ym,Lx,1);

%
nx = 0:(Lx-1);
nm = 0:(size(xm,1)-1);


%%%%%%%%%%%
% figuras
%%%%%%%%%%%

```

```
%
FigInd = 0;
YlabelStrX = ['x_0[n]' ; 'x_1[n]' ; 'x_2[n]' ;
              'x_3[n]' ; 'x_4[n]' ; 'x_5[n]' ];
YlabelStrY = ['y_0[n]' ; 'y_1[n]' ; 'y_2[n]' ;
              'y_3[n]' ; 'y_4[n]' ; 'y_5[n]' ];

%
FigInd = FigInd + 1;
figure(FigInd)
stem(nx,x)
ylabel('x[n]')
xlabel('n')

%
FigInd = FigInd + 1;
figure(FigInd)
%
for SubInd = 1:6
    subplot(2,3,SubInd)
    stem(nm,xm(:,SubInd))
    ylabel(YlabelStrX(SubInd,:))
    xlabel('n')
    AV = axis;
    axis([AV(1) AV(2) -2  2]);
end

%
FigInd = FigInd + 1;
figure(FigInd)
%
for SubInd = 1:6
    subplot(2,3,SubInd)
    stem(nm,ym(:,SubInd))
    ylabel(YlabelStrY(SubInd,:))
    xlabel('n')
    AV = axis;
    axis([AV(1) AV(2) -2  2]);
end

%
FigInd = FigInd + 1;
figure(FigInd)
stem(nx,y)
ylabel('y[n]')
xlabel('n')


%
%  EOF
%
```

Código 3.9: Sequência *x[n]* finita.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: x_seq.m
```

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function s = x_seq(t)

%
% x_seq   Gera sequencia finita x[n].
%

% define o sinal
x0 = 1;
x1 = 2;
x2 = 2;
x3 = 1;
x4 = 0;

% gera sinal com indices originais
st = (x0*(t==0) + x1*(t==1) + x2*(t==2) + x3*(t==3) + x4*(t==4));

% descobre indices nao discretos de t
not_disc_ind = find(round(t) ~= t);

% indica indices que deverao ser ignorados
st(not_disc_ind) = NaN;

% retorna sinal com indices inteiros
s = st';

%
% EOF
%
```

Código 3.10: Sequência *h[n]* finita.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: h_seq.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function s = h_seq(t)

%
% h_seq   Gera sequencia finita h[n].
%

% define o sinal
h0 = 0;
h1 = 1;
h2 = 2;
h3 = 0;
h4 = 0;

% gera sinal com indices originais
st = (h0*(t==0) + h1*(t==1) + h2*(t==2) + h3*(t==3) + h4*(t==4));

% descobre indices nao discretos de t
```

```
not_disc_ind = find(round(t) ~= t);

% indica indices que deverao ser ignorados
st(not_disc_ind) = NaN;

% retorna sinal com indices inteiros
s = st';

%
% EOF
%
```

Código 3.11: Sequência *x[n]* periódica.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: x_seq_per.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function s = x_seq_per(Nf,D,NoP)

%
% x_seq_per    Gera sequencia periodica x[n].
%

% define o sinal
n = 0:(Nf-1);
v = x_seq(n);
if (NoP < 0)
     v = flipud(v);
     v = circshift(v,1);
     NoP = -NoP;
end
vp = [v v v];
vpd = circshift(vp,D);

% retorna sinal com indices inteiros
s = reshape(vpd, (NoP*Nf), 1);

%
% EOF
%
```

Código 3.12: Sequência *h[n]* periódica.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: h_seq_per.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function s = h_seq_per(Nf,D,NoP)

%
```

```
% h_seq_per   Gera sequencia periodica h[n].
%

% define o sinal
n = 0:(Nf-1);
v = h_seq(n);
if (NoP < 0)
    v = flipud(v);
    v = circshift(v,1);
    NoP = -NoP;
end
vp = [v v v];
vpd = circshift(vp,D);

% retorna sinal com indices inteiros
s = reshape(vpd, (NoP*Nf), 1);

%
% EOF
%
```

Código 3.13: Relacionamento das convoluções linear, periódica e circular.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: conv_linear_periodica_circular.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
% Titulo: Demo de relacionamento                                                %
%         das convolucoes                                                       %
%         linear, periodica e circular                                          %
%                                                                               %
% Autor : Alexandre Santos de la Vega                                           %
% Datas : /2012-01/                                                             %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% limpeza do ambiente de trabalho
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%  variaveis
clear all
%  janelas
close all


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Constantes para graficos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
35 %
   FigInd = 0;
37

39 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Constantes para convolucao linear
41 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %
43 N = 7;
   D = 7;
45

47 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % x e h originais
49 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %
51 FigInd = FigInd + 1;
   figure(FigInd)
53
   %
55 n = 0:(N-1);

57 %
   xn = x_seq(n);
59 %
   subplot(2,1,1)
61 stem(n,xn)
   title('Entrada')
63 ylabel('x[n]')
   xlabel('n')
65
   %
67 hn = h_seq(n);
   %
69 subplot(2,1,2)
   stem(n,hn)
71 title('Resposta ao impulso')
   ylabel('h[n]')
73 xlabel('n')

75
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
77 % x espelhados e deslocados
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
79 %
   FigInd = FigInd + 1;
81 figure(FigInd)

83 %
   YlabelStrX = ['x[-k]   ' ; 'x[-k+1]' ; 'x[-k+2]' ;
85               'x[-k+3]' ; 'x[-k+4]' ; 'x[-k+5]' ;
                 'x[-k+6]'
87             ];
   k = -(D-1):(D-1);
89

   %
91 xk = x_seq(k);
   %
93 subplot(2,4,1)
```

```
stem(k,xk)
title('Entrada')
ylabel('x[k]')
xlabel('k')
AV = axis;
axis([-6 6 0 2]);

%
mxd = [];
for d = 0:(D-1)
    mxd = [mxd x_seq(-k + d)];
end
%
for SubInd = 0:(D-1)
    subplot(2,4,(SubInd+2))
    stem(k,mxd(:,(SubInd+1)))
    ylabel(YlabelStrX((SubInd+1),:))
    xlabel('k')
    AV = axis;
    axis([-6 6 0 2]);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% h espelhados e deslocados
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
FigInd = FigInd + 1;
figure(FigInd)

%
YlabelStrH = ['h[-k]   ' ; 'h[-k+1]' ; 'h[-k+2]' ;
              'h[-k+3]' ; 'h[-k+4]' ; 'h[-k+5]' ;
              'h[-k+6]'
             ];
k = -(D-1):(D-1);

%
hk = h_seq(k);
%
subplot(2,4,1)
stem(k,hk)
title('Resposta ao impulso')
ylabel('h[k]')
xlabel('k')
AV = axis;
axis([-6 6 0 2]);

%
mhd = [];
for d = 0:(D-1)
    mhd = [mhd h_seq(-k + d)];
end

%
for SubInd = 0:(D-1)
    subplot(2,4,(SubInd+2))
    stem(k,mhd(:,(SubInd+1)))
    ylabel(YlabelStrH((SubInd+1),:))
```

```
153     xlabel('k')
        AV = axis;
155     axis([-6 6 0 2]);
    end
157

159 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % convolucao linear de x com h espelhados e deslocados
161 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
163 FigInd = FigInd + 1;
    figure(FigInd)
165
    %
167 YlabelStrXHd = ['x[k] \cdot h[-k  ]' ; 'x[k] \cdot h[-k+1]' ;
                   'x[k] \cdot h[-k+2]' ; 'x[k] \cdot h[-k+3]' ;
169                'x[k] \cdot h[-k+4]' ; 'x[k] \cdot h[-k+5]' ;
                   'x[k] \cdot h[-k+6]'
171               ];
    k = -(D-1):(D-1);
173
    %
175 myxhd = [];
    for ind = 1:D
177     myxhd = [myxhd (xk .* mhd(:,ind))];
    end
179 %
    for SubInd = 0:(D-1)
181     subplot(2,4,(SubInd+1))
        stem(k,myxhd(:,(SubInd+1)))
183     ylabel(YlabelStrXHd((SubInd+1),:))
        xlabel('k')
185     AV = axis;
        axis([-6 6 0 6]);
187 end

189 %
    y = [];
191 for ind = 1:D
        y = [y sum(myxhd(:,ind))];
193 end
    %
195 subplot(2,4,8)
    stem(n,y)
197 title('Convolução linear')
    ylabel('y_L[n]')
199 xlabel('n')
    AV = axis;
201 axis([-6 6 0 6]);
    %
203

205 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % convolucao linear de h com x espelhados e deslocados
207 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
209 FigInd = FigInd + 1;
    figure(FigInd)
211
```

```
%
YlabelStrHXd = ['h[k] \cdot x[-k   ]' ; 'h[k] \cdot x[-k+1]' ;
                'h[k] \cdot x[-k+2]' ; 'h[k] \cdot x[-k+3]' ;
                'h[k] \cdot x[-k+4]' ; 'h[k] \cdot x[-k+5]' ;
                'h[k] \cdot x[-k+6]'
               ];
k = -(D-1):(D-1);

%
myhxd = [];
for ind = 1:D
    myhxd = [myhxd (hk .* mxd(:,ind))];
end
%
for SubInd = 0:(D-1)
    subplot(2,4,(SubInd+1))
    stem(k,myhxd(:,(SubInd+1)))
    ylabel(YlabelStrHXd((SubInd+1),:))
    xlabel('k')
    AV = axis;
    axis([-6 6 0 6]);
end

%
y = [];
for ind = 1:D
    y = [y sum(myhxd(:,ind))];
end
%
subplot(2,4,8)
stem(n,y)
title('Convolução linear')
ylabel('y_L[n]')
xlabel('n')
AV = axis;
axis([-6 6 0 6]);
%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Constantes para convolucao periodica
% com numero insuficiente de amostras
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
Np = 4;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% x e h periodicos
% Np = 4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
FigInd = FigInd + 1;
figure(FigInd)

%
np = -Np:(2*Np - 1);
RedLine = (Np+1):(2*Np);
NoP = 3;
```

```

%
xpn = x_seq_per(Np,0,NoP);
%
subplot(2,1,1)
stem(np,xpn)
hold on
stem(np(RedLine),xpn(RedLine),'r')
title('Entrada periódica')
ylabel('x_p[n]')
xlabel('n')

%
hpn = h_seq_per(Np,0,NoP);
%
subplot(2,1,2)
stem(np,hpn)
hold on
stem(np(RedLine),hpn(RedLine),'r')
title('Resposta ao impulso periódica')
ylabel('h_p[n]')
xlabel('n')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% x periodicos espelhados e deslocados
% Np = 4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
FigInd = FigInd + 1;
figure(FigInd)

%
YlabelStrXp = ['x_p[-k]    ' ; 'x_p[-k+1]' ; 'x_p[-k+2]' ;
                'x_p[-k+3]'
               ];
k = np;

%
xpk = xpn;
%
subplot(2,3,1)
stem(np,xpk)
hold on
stem(np(RedLine),xpk(RedLine),'r')
title('Entrada periódica')
ylabel('x_p[k]')
xlabel('k')
AV = axis;
axis([-Np (2*Np - 1) 0 2]);

%
NoP = -3;
mxpd = [];
for d = 0:(Np-1)
    mxpd = [mxpd x_seq_per(Np,d,NoP)];
end
%
for SubInd = 0:(Np-1)
```

```
    subplot(2,3,(SubInd+2))
331     stem(k,mxpd(:,(SubInd+1)))
    hold on
333     stem(k(RedLine),mxpd(RedLine,(SubInd+1)),'r')
    ylabel(YlabelStrXp((SubInd+1),:))
335     xlabel('k')
AV = axis;
337 axis([-Np (2*Np - 1) 0 2]);
end
339

341 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% h periodicos espelhados e deslocados
343 % Np = 4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
345 %
FigInd = FigInd + 1;
347 figure(FigInd)

349 %
YlabelStrHp = ['h_p[-k]   ' ; 'h_p[-k+1]' ; 'h_p[-k+2]' ;
351                'h_p[-k+3]'
                  ];
353 k = np;

355 %
hpk = hpn;
357 %
subplot(2,3,1)
359 stem(np,hpk)
hold on
361 stem(np(RedLine),hpk(RedLine),'r')
title('Resposta ao impulso periódica')
363 ylabel('h_p[k]')
xlabel('k')
365 AV = axis;
axis([-Np (2*Np - 1) 0 2]);
367
%
369 NoP = -3;
mhpd = [];
371 for d = 0:(Np-1)
    mhpd = [mhpd h_seq_per(Np,d,NoP)];
373 end
%
375 for SubInd = 0:(Np-1)
    subplot(2,3,(SubInd+2))
377     stem(k,mhpd(:,(SubInd+1)))
    hold on
379     stem(k(RedLine),mhpd(RedLine,(SubInd+1)),'r')
    ylabel(YlabelStrHp((SubInd+1),:))
381     xlabel('k')
AV = axis;
383 axis([-Np (2*Np - 1) 0 2]);
end
385

387 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% convolucao periodica de x com h espelhados e deslocados
```

```
% Np = 4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
FigInd = FigInd + 1;
figure(FigInd)

%
YlabelStrXpHpd = ['x_p[k] \cdot h_p[-k   ]' ; 'x_p[k] \cdot h_p[-k+1]' ;
                  'x_p[k] \cdot h_p[-k+2]' ; 'x_p[k] \cdot h_p[-k+3]'
                 ];
k = 0:(Np-1);
kind = k+1;

%
myxphpd = [];
for ind = 1:Np
    myxphpd = [myxphpd (xpk(kind) .* mhpd(kind,ind))];
end
%
for SubInd = 0:(Np-1)
    subplot(2,4,(SubInd+1))
    stem(k,myxphpd(kind,(SubInd+1)))
    ylabel(YlabelStrXpHpd((SubInd+1),:))
    xlabel('k')
    AV = axis;
    axis([0 (Np - 1) 0 6]);
end

%
y = [];
for ind = 1:Np
    y = [y sum(myxphpd(kind,ind))];
end
%
subplot(2,4,(Np+4))
stem(k,y)
title('Convolução periódica (N_p = 4)')
ylabel('y_p[n]')
xlabel('n')
AV = axis;
axis([0 (Np - 1) 0 6]);
%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% convolucao periodica de h com x espelhados e deslocados
% Np = 4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
FigInd = FigInd + 1;
figure(FigInd)

%
YlabelStrHpXpd = ['h_p[k] \cdot x_p[-k   ]' ; 'h_p[k] \cdot x_p[-k+1]' ;
                  'h_p[k] \cdot x_p[-k+2]' ; 'h_p[k] \cdot x_p[-k+3]'
                 ];
k = 0:(Np-1);
kind = k+1;

```

```
%
myhpxpd = [];
for ind = 1:Np
    myhpxpd = [myhpxpd (hpk(kind) .* mxpd(kind,ind))];
end
%
for SubInd = 0:(Np-1)
    subplot(2,4,(SubInd+1))
    stem(k,myhpxpd(kind,(SubInd+1)))
    ylabel(YlabelStrHpXpd((SubInd+1),:))
    xlabel('k')
    AV = axis;
    axis([0 (Np - 1) 0 6]);
end

%
y = [];
for ind = 1:Np
    y = [y sum(myhpxpd(kind,ind))];
end
%
subplot(2,4,(Np+4))
stem(k,y)
title('Convolução periódica (N_p = 4)')
ylabel('y_p[n]')
xlabel('n')
AV = axis;
axis([0 (Np - 1) 0 6]);
%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Constantes para convolucao periodica
% com numero suficiente de amostras
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
Np = 6;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% x e h periodicos
% Np = 6
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
FigInd = FigInd + 1;
figure(FigInd)

%
np = -Np:(2*Np - 1);
RedLine = (Np+1):(2*Np);
NoP = 3;

%
xpn = x_seq_per(Np,0,NoP);
%
subplot(2,1,1)
stem(np,xpn)
hold on
stem(np(RedLine),xpn(RedLine),'r')
```

```
title('Entrada periódica')
ylabel('x_p[n]')
xlabel('n')

%
hpn = h_seq_per(Np,0,NoP);
%
subplot(2,1,2)
stem(np,hpn)
hold on
stem(np(RedLine),hpn(RedLine),'r')
title('Resposta ao impulso periódica')
ylabel('h_p[n]')
xlabel('n')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% x periodicos espelhados e deslocados
% Np = 6
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
FigInd = FigInd + 1;
figure(FigInd)

%
YlabelStrXp = ['x_p[-k]   ' ; 'x_p[-k+1]' ; 'x_p[-k+2]' ;
               'x_p[-k+3]' ; 'x_p[-k+4]' ; 'x_p[-k+5]'
              ];
k = np;

%
xpk = xpn;
%
subplot(2,4,1)
stem(np,xpk)
hold on
stem(np(RedLine),xpk(RedLine),'r')
title('Entrada periódica')
ylabel('x_p[k]')
xlabel('k')
AV = axis;
axis([-Np (2*Np - 1) 0 2]);

%
NoP = -3;
mxpd = [];
for d = 0:(Np-1)
    mxpd = [mxpd x_seq_per(Np,d,NoP)];
end
%
for SubInd = 0:(Np-1)
    subplot(2,4,(SubInd+2))
    stem(k,mxpd(:,(SubInd+1)))
    hold on
    stem(k(RedLine),mxpd(RedLine,(SubInd+1)),'r')
    ylabel(YlabelStrXp((SubInd+1),:))
    xlabel('k')
AV = axis;
axis([-Np (2*Np - 1) 0 2]);
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% h periodicos espelhados e deslocados
% Np = 6
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
FigInd = FigInd + 1;
figure(FigInd)

%
YlabelStrHp = ['h_p[-k]   ' ; 'h_p[-k+1]' ; 'h_p[-k+2]' ;
                'h_p[-k+3]' ; 'h_p[-k+4]' ; 'h_p[-k+5]'
               ];
k = np;

%
hpk = hpn;
%
subplot(2,4,1)
stem(np,hpk)
hold on
stem(np(RedLine),hpk(RedLine),'r')
title('Resposta ao impulso periódica')
ylabel('h_p[k]')
xlabel('k')
AV = axis;
axis([-Np (2*Np - 1) 0 2]);

%
NoP = -3;
mhpd = [];
for d = 0:(Np-1)
    mhpd = [mhpd h_seq_per(Np,d,NoP)];
end
%
for SubInd = 0:(Np-1)
    subplot(2,4,(SubInd+2))
    stem(k,mhpd(:,(SubInd+1)))
    hold on
    stem(k(RedLine),mhpd(RedLine,(SubInd+1)),'r')
    ylabel(YlabelStrHp((SubInd+1),:))
    xlabel('k')
AV = axis;
axis([-Np (2*Np - 1) 0 2]);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% convolucao periodica de x com h espelhados e deslocados
% Np = 6
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
FigInd = FigInd + 1;
figure(FigInd)

%
YlabelStrXpHpd = ['x_p[k] \cdot h_p[-k  ]' ; 'x_p[k] \cdot h_p[-k+1]' ;
```

```
625                     'x_p[k] \cdot h_p[-k+2]' ; 'x_p[k] \cdot h_p[-k+3]' ;
                        'x_p[k] \cdot h_p[-k+4]' ; 'x_p[k] \cdot h_p[-k+5]'
627                    ];
k = 0:(Np-1);
629 kind = k+1;

631 %
myxphpd = [];
633 for ind = 1:Np
    myxphpd = [myxphpd (xpk(kind) .* mhpd(kind,ind))];
635 end
%
637 for SubInd = 0:(Np-1)
    subplot(2,4,(SubInd+1))
639     stem(k,myxphpd(kind,(SubInd+1)))
    ylabel(YlabelStrXpHpd((SubInd+1),:))
641     xlabel('k')
    AV = axis;
643     axis([0 (Np - 1) 0 6]);
end
645
%
647 y = [];
for ind = 1:Np
649     y = [y sum(myxphpd(kind,ind))];
end
651 %
subplot(2,4,(Np+2))
653 stem(k,y)
title('Convolução periódica (N_p = 6)')
655 ylabel('y_p[n]')
xlabel('n')
657 AV = axis;
axis([0 (Np - 1) 0 6]);
659 %


661
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
663 % convolucao periodica de h com x espelhados e deslocados
% Np = 6
665 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
667 FigInd = FigInd + 1;
figure(FigInd)
669
%
671 YlabelStrHpXpd = ['h_p[k] \cdot x_p[-k  ]' ; 'h_p[k] \cdot x_p[-k+1]' ;
                    'h_p[k] \cdot x_p[-k+2]' ; 'h_p[k] \cdot x_p[-k+3]' ;
673                 'h_p[k] \cdot x_p[-k+4]' ; 'h_p[k] \cdot x_p[-k+5]'
                   ];
675 k = 0:(Np-1);
kind = k+1;
677
%
679 myhpxpd = [];
for ind = 1:Np
681     myhpxpd = [myhpxpd (hpk(kind) .* mxpd(kind,ind))];
end
683 %
```

```
for SubInd = 0:(Np-1)
    subplot(2,4,(SubInd+1))
    stem(k,myhpxpd(kind,(SubInd+1)))
    ylabel(YlabelStrHpXpd((SubInd+1),:))
    xlabel('k')
    AV = axis;
    axis([0 (Np - 1) 0 6]);
end

%
y = [];
for ind = 1:Np
    y = [y sum(myhpxpd(kind,ind))];
end
%
subplot(2,4,(Np+2))
stem(k,y)
title('Convolução periódica (N_p = 6)')
ylabel('y_p[n]')
xlabel('n')
AV = axis;
axis([0 (Np - 1) 0 6]);
%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Constantes para convolucao periodica
% com numero insuficiente de amostras
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
Nc = 4; % Nesse exercicio, usar 2 < N < 8.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% convolucao circular de x com h circulante
% Nc = 4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
FigInd = FigInd + 1;
figure(FigInd)

%
ind = 1:Nc;
n = (ind-1);

%
xc = xn(ind);
%
hc = hn(ind);
c = hc;
for i = 2:Nc
    hc = circshift(hc,[1 0]);
    c = [c hc];
end
%
yc = c*xc;
%
stem(n,yc)
title('Convolução circular com h[n] circulante (N_c = 4)')
```

```
743 ylabel('y_c[n]')
    xlabel('n')
745 AV = axis;
    axis([0 5 0 6]);
747

749 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % convolucao circular de h com x circulante
751 % Nc = 4
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
753 %
    FigInd = FigInd + 1;
755 figure(FigInd)

757 %
    ind = 1:Nc;
759 n = (ind-1);

761 %
    hc = hn(ind);
763 %
    xc = xn(ind);
765 c = xc;
    for i = 2:Nc
767     xc = circshift(xc,[1 0]);
        c = [c xc];
769 end
    %
771 yc = c*hc;
    %
773 stem(n,yc)
    title('Convolução circular com x[n] circulante (N_c = 4)')
775 ylabel('y_c[n]')
    xlabel('n')
777 AV = axis;
    axis([0 5 0 6]);
779

781 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Constantes para convolucao periodica
783 % com numero suficiente de amostras
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
785 %
    Nc = 6; % Nesse exercicio, usar 2 < N < 8.
787

789 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % convolucao circular de x com h circulante
791 % Nc = 6
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
793 %
    FigInd = FigInd + 1;
795 figure(FigInd)

797 %
    ind = 1:Nc;
799 n = (ind-1);

801 %
```

```
xc = xn(ind);
%
hc = hn(ind);
c = hc;
for i = 2:Nc
    hc = circshift(hc,[1 0]);
    c = [c hc];
end
%
yc = c*xc;
%
stem(n,yc)
title('Convolução circular com h[n] circulante (N_c = 6)')
ylabel('y_c[n]')
xlabel('n')
AV = axis;
axis([0 5 0 6]);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% convolucao circular de h com x circulante
% Nc = 6
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
FigInd = FigInd + 1;
figure(FigInd)

%
ind = 1:Nc;
n = (ind-1);

%
hc = hn(ind);
%
xc = xn(ind);
c = xc;
for i = 2:Nc
    xc = circshift(xc,[1 0]);
    c = [c xc];
end
%
yc = c*hc;
%
stem(n,yc)
title('Convolução circular com x[n] circulante (N_c = 6)')
ylabel('y_c[n]')
xlabel('n')
AV = axis;
axis([0 5 0 6]);

%
% EOF
%
```

3.4 Sequências mais comumente empregadas

- O Código 3.14 apresenta um menu de seleção para sequências básicas.
- O Código 3.15 apresenta a geração da função degrau unitário discreto.
- O Código 3.16 apresenta a geração da função Dirichlet unitária.
- O Código 3.17 apresenta a geração da função *gate* retangular unitário discreto.
- O Código 3.18 apresenta a geração da função impulso unitário discreto.
- O Código 3.19 apresenta a geração da função linear unitária discreta.
- O Código 3.20 apresenta a geração da função módulo unitário discreto.
- O Código 3.21 apresenta a geração da função potenciação unitária discreta.
- O Código 3.22 apresenta a geração da função rampa unitária discreta.
- O Código 3.23 apresenta a geração da função *Signum* unitário discreto.
- O Código 3.24 apresenta a geração da função *Sinc* unitária.
- O Código 3.25 apresenta uma demonstração de sequência exponencial real.

Código 3.14: Menu de seleção para sequências básicas.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: seq_bas_menu.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
clear all
close all

menu_choice = 1;
while (menu_choice ~= 0)
    for i = 1:50
        disp(' ')
    end
    disp('      *********************************')
    disp('      * Geração de sequências básicas *')
    disp('      *********************************')
    disp(' ')
    disp('  Opções:')
    disp('           00 - Sair')
    disp('           01 - Degrau unitário')
    disp('           02 - Dirichlet unitária')
    disp('           03 - Gate retangular unitário')
    disp('           04 - Impulso unitário')
    disp('           05 - Sequência linear unitária')
    disp('           06 - Módulo unitário')
    disp('           07 - Exponencial unitária')
    disp('           08 - Rampa unitária')
```

```
    disp('          09 - Signum unitário')
    disp('          10 - Sinc unitária')
    disp(' ')
    menu_choice = input('  Escolha: ');
    disp(' ')

    switch menu_choice
        case {0}
            disp(' ')
            disp('  Bye...')
            disp(' ')
        case {1}
            disp('  Degrau unitário')
            t = input('  Faixa de tempo no formato INÍCIO:FIM = ');
            stem(t,degrau_unitario(t))
            title('Degrau unitário')
            ylabel('u [n]')
            xlabel('n')
        case {2}
            disp('  Dirichlet unitária')
            N = input('  Período = ');
            t = input('  Faixa de tempo no formato INÍCIO:FIM = ');
            [s,n] = drcl_unitaria(t,N);
            stem(t,s)
            title('Dirichlet unitária')
            ylabel('Drcl [n]')
            xlabel('n')
        case {3}
            disp('  Gate retangular unitário')
            Ng = input('  Número de amostras do gate = ');
            t  = input('  Faixa de tempo no formato  inicio:fim =   ');
            stem(t,gate_retang_unitario(Ng,t))
            title('Gate retangular unitário')
            ylabel('G_{Ng} [n]')
            xlabel('n')
        case {4}
            disp('  Impulso unitário')
            t = input('  Faixa de tempo no formato INÍCIO:FIM = ');
            stem(t,impulso_unitario(t))
            title('Impulso unitário')
            ylabel('\delta [n]')
            xlabel('n')
        case {5}
            disp('  Sequência linear unitária')
            t = input('  Faixa de tempo no formato INÍCIO:FIM = ');
            stem(t,linear_unitaria(t))
            title('Sequência linear unitária')
            ylabel('Lin [n]')
            xlabel('n')
        case {6}
            disp('  Módulo unitário')
            t = input('  Faixa de tempo no formato INÍCIO:FIM = ');
```

```
            stem(t,modulo_unitario(t))
84          title('Módulo unitário')
            ylabel('Mod [n]')
86          xlabel('n')
        case {7}
88          disp('  Exponencial unitária')
            base = input('  Base = ');
90          t    = input('  Faixa de tempo no formato  inicio:fim
    =    ');
            stem(t,power_unitaria(base,t))
92          title('Exponencial unitária')
            ylabel('Base^n')
94          xlabel('n')
        case {8}
96          disp('  Rampa unitária')
            t = input('  Faixa de tempo no formato INÍCIO:FIM =
    ');
98          stem(t,rampa_unitaria(t))
            title('Rampa unitária')
100         ylabel('Rmp [n]')
            xlabel('n')
102     case {9}
            disp('  Signum unitário')
104         t = input('  Faixa de tempo no formato INÍCIO:FIM =
    ');
            stem(t,signum_unitario(t))
106         title('Signum unitário')
            ylabel('Sgn [n]')
108         xlabel('n')
        case {10}
110         disp('  Sinc unitária')
            t = input('  Faixa de tempo no formato INÍCIO:FIM =
    ');
112         [s,n] = sinc_unitaria(t);
            stem(t,s)
114         title('Sinc unitária')
            ylabel('Sinc [n]')
116         xlabel('n')
        otherwise
118 %           'do nothing...'
    end  % switch
120 end  % while

122

    %
124 % EOF
    %
```

Código 3.15: Geração da função degrau unitário discreto.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %
3  % Arquivo: degrau_unitario.m
   %
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %
7  function s = degrau_unitario(t)
```

```
%
% degrau_unitario    Gera funcao Degrau Unitario Digital.
%

% gera sinal com indices originais
st = 1.*(t>=0);

% descobre indices nao discretos de t
not_disc_ind = find(round(t) ~= t);

% indica indices que deverao ser ignorados
st(not_disc_ind) = NaN;

% retorna sinal com indices inteiros
s = st;

%
% EOF
%
```

Código 3.16: Geração da função Dirichlet unitária.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: drcl_unitaria.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [s,n] = drcl_unitaria(t,N)

%
% drcl_unitaria    Gera função Dirichlet Unitária.
%
%     É esperado: t = -Tmax ... 0 ... Tmax.
%

% descobre indice t=0
ind_t_0 = find(t == 0);

% gera nova faixa de indices
n = (1:length(t)) - ind_t_0;

% gera sinal com indices originais
s = diric(2*pi*t,N);

%
% EOF
%
```

Código 3.17: Geração da função *gate* retangular unitário discreto.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: gate_retang_unitario.m
```

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function s = gate_retang_unitario(Ng, t)

%
% gate_retang_unitario   Gera funcao Gate Retangular Unitario Digital.
%

% gera sinal com indices originais
st = 1.*(- abs(Ng)<=t & t<= abs(Ng));

% descobre indices nao discretos de t
not_disc_ind = find(round(t) ~= t);

% indica indices que deverao ser ignorados
st(not_disc_ind) = NaN;

% retorna sinal com indices inteiros
s = st;

%
% EOF
%
```

Código 3.18: Geração da função impulso unitário discreto.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: impulso_unitario.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function s = impulso_unitario(t)

%
% impulso_unitario   Gera funcao Impulso Unitario Digital.
%

% gera sinal com indices originais
st = 1.*(t==0);

% descobre indices nao discretos de t
not_disc_ind = find(round(t) ~= t);

% indica indices que deverao ser ignorados
st(not_disc_ind) = NaN;

% retorna sinal com indices inteiros
s = st;

%
% EOF
%
```

Código 3.19: Geração da função linear unitária discreta.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: linear_unitaria.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function s = linear_unitaria(t)

%
% linear_unitaria   Gera funcao Linear Unitaria Digital .
%

% gera sinal com indices originais
st = t;

% descobre indices nao discretos de t
not_disc_ind = find(round(t) ~= t);

% indica indices que deverao ser ignorados
st(not_disc_ind) = NaN;

% retorna sinal com indices inteiros
s = st;

%
% EOF
%
```

Código 3.20: Geração da função módulo unitário discreto.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: modulo_unitario.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function s = modulo_unitario(t)

%
% modulo_unitario   Gera funcao Modulo Unitario Digital .
%

% gera sinal com indices originais
st = -t.*(t<0) + t.*(0<=t);

% descobre indices nao discretos de t
not_disc_ind = find(round(t) ~= t);

% indica indices que deverao ser ignorados
st(not_disc_ind) = NaN;

% retorna sinal com indices inteiros
s = st;

%
% EOF
```

```
27 %
```

Código 3.21: Geração da função potenciação unitária discreta.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %
3  % Arquivo: power_unitaria.m
   %
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %
7  function s = power_unitaria(base, t)

9  %
   % power_unitaria   Gera funcao Potenciacao Unitaria Digital .
11 %

13 % gera sinal com indices originais
   st = power(base,t);
15
   % descobre indices nao discretos de t
17 not_disc_ind = find(round(t) ~= t);

19 % indica indices que deverao ser ignorados
   st(not_disc_ind) = NaN;
21
   % retorna sinal com indices inteiros
23 s = st;

25 %
   % EOF
27 %
```

Código 3.22: Geração da função rampa unitária discreta.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %
3  % Arquivo: rampa_unitaria.m
   %
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %
7  function s = rampa_unitaria(t)

9  %
   % rampa_unitaria   Gera funcao Rampa Unitaria Digital .
11 %

13 % gera sinal com indices originais
   st = t.*(t>=0);
15
   % descobre indices nao discretos de t
17 not_disc_ind = find(round(t) ~= t);

19 % indica indices que deverao ser ignorados
   st(not_disc_ind) = NaN;
21
```

```
% retorna sinal com indices inteiros
s = st;

%
% EOF
%
```

Código 3.23: Geração da função *Signum* unitário discreto.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: signum_unitario.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function s = signum_unitario(t)

%
% signum_unitario   Gera funcao Signum Unitario Digital .
%

% gera sinal com indices originais
st = -1.*(t<0) + 1.*(0<t);

% descobre indices nao discretos de t
not_disc_ind = find(round(t) ~= t);

% indica indices que deverao ser ignorados
st(not_disc_ind) = NaN;

% retorna sinal com indices inteiros
s = st;

%
% EOF
%
```

Código 3.24: Geração da função *Sinc* unitária.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: sinc_unitaria.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [s,n] = sinc_unitaria(t)

%
% sinc_unitaria   Gera funcao Sinc Unitaria.
%

% descobre indice t=0
ind_t_0 = find(t == 0);

% gera nova faixa de indices
```

```
n = (1:length(t)) - ind_t_0;

% gera sinal com indices originais
s = sinc(t);

%
% EOF
%
```

Código 3.25: Demonstração de sequência exponencial real.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: power_real_demo.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Titulo: Demo de sequencia exponencial real
%
% Autor : Alexandre Santos de la Vega
% Data  : 30/03/2k9 ; 08/09/2k9
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

% limpeza do ambiente de trabalho
%  variaveis
clear all
%  janelas
close all


% definicao dos parametros
n = 0:10;
A = 1;
alfa_pos_1_inf =   2;
alfa_pos_1     =   1;
alfa_pos_0_1   =   0.2;
alfa_neg_1_0   = - 0.2;
alfa_neg_1     = - 1;
alfa_neg_inf_1 = - 2;


% graficos
figure(1)
stem(n, (A*power_unitaria(alfa_pos_1_inf,n)))
title('exponencial real: \alpha^n , \alpha = (1 ; \infty)')
xlabel('n')

figure(2)
stem(n, (A*power_unitaria(alfa_pos_1,n)))
title('exponencial real: \alpha^n , \alpha = 1')
```

```
xlabel('n')

figure(3)
stem(n, (A*power_unitaria(alfa_pos_0_1,n)))
title('exponencial real: \alpha^n , \alpha = (0 ; 1)')
xlabel('n')

figure(4)
stem(n, (A*power_unitaria(alfa_neg_1_0,n)))
title('exponencial real: \alpha^n , \alpha = (-1 ; 0)')
xlabel('n')

figure(5)
stem(n, (A*power_unitaria(alfa_neg_1,n)))
title('exponencial real: \alpha^n , \alpha = -1')
xlabel('n')

figure(6)
stem(n, (A*power_unitaria(alfa_neg_inf_1,n)))
title('exponencial real: \alpha^n , \alpha = (-\infty ; -1)')
xlabel('n')


figure(7)
subplot(2,3,1)
stem(n, (A*power_unitaria(alfa_pos_1_inf,n)))
ylabel('\alpha = (1 ; \infty)')
xlabel('n')

subplot(2,3,2)
stem(n, (A*power_unitaria(alfa_pos_1,n)))
ylabel('\alpha = 1')
xlabel('n')

title('exponencial real: \alpha^n')

subplot(2,3,3)
stem(n, (A*power_unitaria(alfa_pos_0_1,n)))
ylabel('\alpha = (0 ; 1)')
xlabel('n')

subplot(2,3,6)
stem(n, (A*power_unitaria(alfa_neg_1_0,n)))
ylabel('\alpha = (-1 ; 0)')
xlabel('n')

subplot(2,3,5)
stem(n, (A*power_unitaria(alfa_neg_1,n)))
ylabel('\alpha = -1')
xlabel('n')

subplot(2,3,4)
stem(n, (A*power_unitaria(alfa_neg_inf_1,n)))
ylabel('\alpha = (-\infty ; -1)')
xlabel('n')


%
% EOF
```

```
106 |%
```

Capítulo 4

Sequências exponenciais

4.1 Introdução

Sequências exponenciais desempenham um papel fundamental em processamento de sinais.

Nas seções que se seguem, são apresentadas diversas listagens de programas, relativas a tal assunto.

4.2 Características relevantes de exponenciais

- O Código 4.1 apresenta a simulação de uma exponencial, passo a passo, interpretada como fasor discreto.
- O Código 4.2 apresenta a associação entre uma exponencial e um fasor.
- O Código 4.3 apresenta a demonstração de um cosseno.
- O Código 4.4 apresenta a demonstração da periodicidade de cosseno discreto.
- O Código 4.5 apresenta a demonstração da periodicidade de cosseno amostrado.

Código 4.1: Exponencial interpretada como fasor discreto.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Arquivo original: fasor_discreto_2021_12_06.m
%
% Arquivo: exp_visao_fasorial.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Titulo: Demo de
%          sequencia exponencial
%          interpretada como fasor discreto.
%
%          - Exp() como fasor.
%          - Relacao de Euler:
%              exp() = cos() + j sin().
%          - Simulacao passo a passo.
```

```
20 %          - Visoes 2D e 3D para exp().
   %          - Taxa de variacao X Nf e Omega.
22 %
   %
24 % Autor : Alexandre Santos de la Vega
   % Data  : / 2021-01 /
26 %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28 %

30 %
   % ---------- ---------- ---------- ---------- ----------
32 %

34 %
   clear all
36 close all

38 %
   % ---------- ---------- ---------- ---------- ----------
40 %

42 %
   % Definicoes gerais
44 %

46 %
   mv_opt = questdlg ("Temporizado ou Manual?",...
48                    "Controle do movimento de fasor discreto",...
       "Temporizado", "Manual (teclado ou mouse)", "Cancela",...
50                    "Temporizado");
   %
52 if(strcmp (mv_opt, "Cancela"))
     %disp ("You cancelled.");
54   return;
   endif
56 %
   if(strcmp (mv_opt, "Temporizado"))
58   %
     options_cell = {"0.0", "0.5", "1.0", "1.5", "2.0", "2.5", "3.0"};
60   [sel_ind, ok] = listdlg (
                               "Name", "Intervalo de tempo (s)",
62                             "CancelString", "Cancela",
                               "ListSize", [180,130],
64                             "ListString", options_cell,
                               "SelectionMode", "Single"
66                            );
     %
68   if (ok == 1)
       t_pause = str2num(options_cell{sel_ind});
70   else
       %disp ("You cancelled.");
72     return;
     endif
74   %
   endif
76
   %
78 options_cell = {"32", "16", "8", "4", "3"};
```

```
[sel_ind, ok] = listdlg (
                          "Name", "Período fundamental (Nf)",
                          "CancelString", "Cancela",
                          "ListSize", [220,100],
                          "ListString", options_cell,
                          "SelectionMode", "Single"
                         );
%
if (ok == 1)
  Nf = str2num(options_cell{sel_ind});
else
  %disp ("You cancelled.");
  return;
endif
%
Omega_Nf = (2*pi/Nf);

%
max_Nf = 0;
for ind=1:length(options_cell)
  Nf_ind = str2num(options_cell{ind});
  if (Nf_ind > max_Nf)
    max_Nf = Nf_ind;
  endif
endfor

%
% ---------- ---------- ---------- ---------- ----------
%

%
% Construcao de
%   Lugar Geometrico com raio constante
%   ou
%   Circulo de raio r
%   ou
%   |z| = r
%   ou
%   z = e^(j Omega)   ;  -oo < Omega < oo
%

%
N = 360;
Omega_step = 2*pi/N;
Omega = 0:Omega_step:(2*pi);

%
circ_1 = e.^(j*Omega);
re_circ_1 = real(circ_1);
im_circ_1 = imag(circ_1);

%
% ---------- ---------- ---------- ---------- ----------
%

%
FigNbr = 0;
%
r_lim = 1.3;
```

```
138 c_lim = ((2*max_Nf)-1);
    %
140 black_val = 0;

142 %
    % ---------- ----------
144 %

146 %
    FigNbr = FigNbr + 1;
148 figure(FigNbr)
    %
150 for n = 0:(Nf-1)
      %
152   f_r1 = e.^(j*Omega_Nf*n);
      re_f_r1 = real(f_r1);
154   im_f_r1 = imag(f_r1);

156   %
      subplot(2,2,1)
158   %
      hold off
160   %
      plot(re_circ_1,im_circ_1,"k")
162   %
      hold on
164   %
      plot(re_f_r1, im_f_r1, "ob", "markersize", 7)
166   plot([0,re_f_r1], [0,im_f_r1], "b")
      %
168   plot(re_f_r1, 0, "or", "markersize", 4)
      plot([0,re_f_r1], [0,0], "r")
170   %
      plot(0, im_f_r1, "og", "markersize", 4)
172   plot([0, 0], [im_f_r1,0], "g")
      %
174   if (re_f_r1 > 0)
        x_txt = 1.1*re_f_r1;
176     y_txt = 1.1*im_f_r1;
      elseif (abs(im_f_r1) < eps)
178     x_txt = 0.9*re_f_r1;
        y_txt = 0.1;
180   else
        x_txt = 0.9*re_f_r1;
182     y_txt = 1.1*im_f_r1;
      end
184   text(x_txt, y_txt, ['n = ', num2str(n)]);
      %
186   axis([-r_lim, r_lim, -r_lim, r_lim])
      % axis("equal") = set(gca,'DataAspectRatio',[1, 1, 1])
188   axis("equal")
      %
190   grid on
      %
192   tit_str_1 = 'Fasor discreto:   z[n] = e^{j(\Omega_{0}n)}';
      tit_str_2 = ['\Omega_{0} = 2\pi/N_{f} = 2\pi/',...
194                num2str(Nf),...
                   ' = ', num2str(Omega_Nf), ' rad'];
196   title ({tit_str_1, tit_str_2});
```

```
    ylabel('y[n] = Im \{z[n]\} = sin(\Omega_{0}n)')
198 xlabel('x[n] = Re \{z[n]\} = cos(\Omega_{0}n)')
    %
200
    %
202 h_sin = subplot(2,2,2);
    %
204 hold on
    %
206 stem(n, im_f_r1, "og")
    %
208 %  axis([0, (Nf-1), -r_lim, r_lim])
    axis([0, c_lim, -r_lim, r_lim])
210 axis("square")
    %
212 grid on
    %
214 title ('y[n] = sin(\Omega_{0}n)');
    ylabel('y[n]')
216 xlabel('n')
    %
218
    %
220 h_cos = subplot(2,2,3);
    %
222 hold on
    %
224 stem(n, re_f_r1, "or")
    %
226 %  axis([0, (Nf-1), -r_lim, r_lim])
    axis([0, c_lim, -r_lim, r_lim])
228 axis("square")
    %
230 grid on
    %
232 title ('x[n] = cos(\Omega_{0}n)');
    ylabel('x[n]')
234 xlabel('n')
    %
236 %view(AZIMUTH, ELEVATION)
    view(90,90)
238 %

240 %
    subplot(2,2,4)
242 %
    hold on
244 %
    % stem3(x,y,z)
246 stem3(re_f_r1, im_f_r1, n, "b")
    %
248 axis([-r_lim, r_lim, -r_lim, r_lim, 0, (Nf-1)])
    axis("square")
250 %
    grid on
252 %
    title('Curva discreta 3D:  z[n] = e^{j (\Omega_0 n)}')
254 zlabel('n')
    ylabel('y[n]')
```

```
256   xlabel('x[n]')
      %
258   %view(AZIMUTH, ELEVATION)
      view(26,18)
260   %
      % fundo preto
262   %set(gca,'color',[black_val,black_val,black_val])
      %
264
      %
266   if(strcmp (mv_opt, "Temporizado"))
        pause(t_pause);
268   else
        b = waitforbuttonpress();
270   endif
      %
272 end
    %
274
    %
276 %msgbox("Fim do período fundamental.", "Movimento do fasor", "warn");
    cont_opt = questdlg("Fim do período fundamental.", "Movimento do fasor",...
278                     "Continua", "Cancela", "Continua");
    %
280 if(strcmp (cont_opt, "Cancela"))
      %disp ("You cancelled.");
282   return;
    endif
284
    %
286 % ---------- ---------- ---------- ---------- ----------
    %
288
    %
290 % completa as projecoes sin() e cos()
    %   apos o periodo fundamental
292 %

294 %
    n = 0:c_lim;
296 %
    f_r1 = e.^(j*Omega_Nf*n);
298 %
    re_f_r1 = real(f_r1);
300 im_f_r1 = imag(f_r1);

302 %
    stem(h_sin, n((Nf+1):end), im_f_r1((Nf+1):end), "ok")
304 stem(h_cos, n((Nf+1):end), re_f_r1((Nf+1):end), "ok")

306 %
    % EOF
308 %
```

Código 4.2: Associação entre exponencial e fasor.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: associa_exp_fasor.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Titulo: Demo de
%         sequencia exponencial
%         interpretada como fasor
%         (visao 3D)
%
% Autor : Alexandre Santos de la Vega
% Data  : / 2010-02 / 2011-01 /
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


% limpeza do ambiente de trabalho
%  variaveis
clear all
%  janelas
close all


% definicao dos parametros
NOD = 7; Theta0 = 0;
%NOD = 7; Theta0 = (2*pi)/((NOD+1)*2);
%NOD = 15; Theta0 = 0;
%NOD = 15; Theta0 = (2*pi)/((NOD+1)*2);
%
Omega0 = (2*pi)/(NOD+1);


% graficos
%
figure(1)
%
% cores
c_eixo = 'k';
c_disc = 'r';
c_cont = 'b';
%
% eixo
x = [0  0];
y = [0  0];
z = [0 NOD];
plot3(z,x,y,c_eixo)
hold on
%
% fasor discreto
x = [0  0];
y = [0  0];
```

```
for n=0:NOD
    x(2) = cos(Omega0*n + Theta0);
    y(2) = sin(Omega0*n + Theta0);
    z    = [n n];
    plot3(z,x,y,c_disc)
end
%
% fasor continuo
n  = (0:0.1:NOD);
x = cos(Omega0*n + Theta0);
y = sin(Omega0*n + Theta0);
z = n;
plot3(z,x,y,c_cont)
%
title('Sequência exponencial interpretada com fasor:
      x[n] = e^{j (\Omega n + \Theta_0)}')
ylabel('cos(\Omega_0 n + \Theta_0)')
zlabel('sin(\Omega_0 n + \Theta_0)')
xlabel('n')
%
grid on
axis square
view(8,12)


%
% EOF
%
```

Código 4.3: Demonstração de cosseno.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: cos_demo.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Titulo: Demo de sequencia senoidal
%
% Autor : Alexandre Santos de la Vega
% Data  : 30/03/2k9 ; 03/09/2k9
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

% limpeza do ambiente de trabalho
%  variaveis
clear all
%  janelas
close all


% definicao dos parametros
```

```
n = 0:50;
A = 1;
% 1o quadrante
Omega11 = (0); % ( = 2*pi ) (16 * Omega12) ( 8 * Omega12) ( 4 * Omega14)
Omega12 = (2*pi)/16;
Omega13 = (2*pi)/8;        % ( 2 * Omega12)
Omega14 = (2*pi)/4;        % ( 2 * Omega13)
% 2o quadrante
Omega21 = (2*pi)/4;
Omega22 = (2*pi)/(8/3);    % ( 3 * Omega13)
Omega23 = (2*pi)/(16/7);   % ( 7 * Omega12)
Omega24 = (2*pi)/2;
% 3o quadrante
Omega31 = (2*pi)/2;
Omega32 = (2*pi)/(16/9);   % ( 9 * Omega12)
Omega33 = (2*pi)/(8/5);    % ( 5 * Omega13)
Omega34 = (2*pi)/(4/3);
% 4o quadrante
Omega41 = (2*pi)/(4/3);
Omega42 = (2*pi)/(8/7);    % ( 7 * Omega13)
Omega43 = (2*pi)/(16/15); % (15 * Omega12)
Omega44 = (2*pi);

FigCntr = 0;


% grafico de angulos OmegaXY
k=1;
z=[];
p=[
    cos(Omega11) + j*sin(Omega11), cos(Omega11) - j*sin(Omega11), ...
    cos(Omega12) + j*sin(Omega12), cos(Omega12) - j*sin(Omega12), ...
    cos(Omega13) + j*sin(Omega13), cos(Omega13) - j*sin(Omega13), ...
    cos(Omega14) + j*sin(Omega14), cos(Omega14) - j*sin(Omega14), ...
    cos(Omega21) + j*sin(Omega21), cos(Omega21) - j*sin(Omega21), ...
    cos(Omega22) + j*sin(Omega22), cos(Omega22) - j*sin(Omega22), ...
    cos(Omega23) + j*sin(Omega23), cos(Omega23) - j*sin(Omega23), ...
    cos(Omega24) + j*sin(Omega24), cos(Omega24) - j*sin(Omega24), ...
    cos(Omega31) + j*sin(Omega31), cos(Omega31) - j*sin(Omega31), ...
    cos(Omega32) + j*sin(Omega32), cos(Omega32) - j*sin(Omega32), ...
    cos(Omega33) + j*sin(Omega33), cos(Omega33) - j*sin(Omega33), ...
    cos(Omega34) + j*sin(Omega34), cos(Omega34) - j*sin(Omega34), ...
    cos(Omega41) + j*sin(Omega41), cos(Omega41) - j*sin(Omega41), ...
    cos(Omega42) + j*sin(Omega42), cos(Omega42) - j*sin(Omega42), ...
    cos(Omega43) + j*sin(Omega43), cos(Omega43) - j*sin(Omega43), ...
    cos(Omega44) + j*sin(Omega44), cos(Omega44) - j*sin(Omega44), ...
   ];

FigCntr = (FigCntr + 1);
figure(FigCntr)
pzmap(zpk(z,p,k))
title('Conjunto de ângulos utilizados no exemplo');


% graficos

% 1o quadrante
FigCntr = (FigCntr + 1);
figure(FigCntr)
```

```
%
subplot(4,1,1)
stem(n, (A*cos(Omega11*n)) )
%
hold on
x = (0:2);
y = (A*cos(Omega11*x));
stem(x, y, 'y' )
hold off
%
ylabel('\Omega = 0')
title('Sinal senoidal: A*cos( \Omega n), \Omega \in [0, \pi / 2]')

subplot(4,1,2)
stem(n, (A*cos(Omega12*n)) )
%
hold on
na = 0:.1:50;
plot(na, (A*cos(Omega12*na)),'k')
hold off
%
hold on
x = (0:2:25);
y = (A*cos(Omega12*x));
stem(x, y, 'r' )
hold off
%
ylabel('\Omega = (2*pi)/16')

subplot(4,1,3)
stem(n, (A*cos(Omega13*n)) )
%
hold on
na = 0:.1:50;
plot(na, (A*cos(Omega13*na)),'k')
hold off
%
hold on
x = (0:(length(y)-1));
stem(x, y, 'r' )
hold off
%
hold on
x = (32:2:50);
y = (A*cos(Omega13*x));
stem(x, y, 'g' )
hold off
%
ylabel('\Omega = (2*pi)/8')

subplot(4,1,4)
stem(n, (A*cos(Omega14*n)) )
%
hold on
na = 0:.1:50;
plot(na, (A*cos(Omega14*na)),'k')
hold off
%
hold on
```

```
145 x = (16:(16+(length(y)-1)));
    stem(x, y, 'g' )
147 hold off
    %
149 %
    hold on
151 x = (0:4:8);
    y = (A*cos(Omega14*x));
153 stem(x, y, 'y' )
    hold off
155 %
    ylabel('\Omega = (2*pi)/4')
157

159 % 2o quadrante
    FigCntr = (FigCntr + 1);
161 figure(FigCntr)
    %
163 subplot(4,1,1)
    stem(n, (A*cos(Omega21*n)) )
165 %
    hold on
167 na = 0:.1:50;
    plot(na, (A*cos(Omega21*na)),'k')
169 hold off
    %
171 ylabel('\Omega = (2*pi)/4')
    title('Sinal senoidal: A*cos( \Omega n), \Omega \in [\pi / 2 , \pi]')
173
    subplot(4,1,2)
175 stem(n, (A*cos(Omega22*n)) )
    %
177 hold on
    na = 0:.1:50;
179 plot(na, (A*cos(Omega22*na)),'k')
    hold off
181 %
    ylabel('\Omega = (2*pi)/(8/3)')
183
    subplot(4,1,3)
185 stem(n, (A*cos(Omega23*n)) )
    %
187 hold on
    na = 0:.1:50;
189 plot(na, (A*cos(Omega23*na)),'k')
    hold off
191 %
    ylabel('\Omega = (2*pi)/(16/7)')
193
    subplot(4,1,4)
195 stem(n, (A*cos(Omega24*n)) )
    %
197 hold on
    na = 0:.1:50;
199 plot(na, (A*cos(Omega24*na)),'k')
    hold off
201 %
    ylabel('\Omega = (2*pi)/2')
203
```

```
205 % 3o quadrante
    FigCntr = (FigCntr + 1);
207 figure(FigCntr)
    %
209 subplot(4,1,1)
    stem(n, (A*cos(Omega31*n)) )
211 %
    ylabel('\Omega = (2*pi)/2')
213 title('Sinal senoidal: A*cos( \Omega n), \Omega \in [\pi , 3 \pi / 2]')

215 subplot(4,1,2)
    stem(n, (A*cos(Omega32*n)) )
217 %
    ylabel('\Omega = (2*pi)/(16/9)')
219
    subplot(4,1,3)
221 stem(n, (A*cos(Omega33*n)) )
    %
223 ylabel('\Omega = (2*pi)/(8/5)')

225 subplot(4,1,4)
    stem(n, (A*cos(Omega34*n)) )
227 %
    ylabel('\Omega = (2*pi)/(4/3)')
229

231 % 4o quadrante
    FigCntr = (FigCntr + 1);
233 figure(FigCntr)
    %
235 subplot(4,1,1)
    stem(n, (A*cos(Omega41*n)) )
237 %
    ylabel('\Omega = (2*pi)/(4/3)')
239 title('Sinal senoidal: A*cos( \Omega n), \Omega \in [3 \pi / 2 , 2 \pi]')

241 subplot(4,1,2)
    stem(n, (A*cos(Omega42*n)) )
243 %
    ylabel('\Omega = (2*pi)/1(8/7)')
245
    subplot(4,1,3)
247 stem(n, (A*cos(Omega43*n)) )
    %
249 ylabel('\Omega = (2*pi)/(16/15)')

251 subplot(4,1,4)
    stem(n, (A*cos(Omega44*n)) )
253 %
    ylabel('\Omega = (2*pi)')
255

257 % comparacoes
    FigCntr = (FigCntr + 1);
259 figure(FigCntr)
    %
261 subplot(4,1,1)
    stem(n, (A*cos(Omega12*n)) )
```

```
263 %
    hold on
265 x = (0:7:50);
    y = (A*cos(Omega12*x));
267 stem(x, y, 'r')
    hold off
269 %
    hold on
271 x = (0:9:50);
    z = (A*cos(Omega12*x));
273 stem(x, z, 'g')
    hold off
275 %
    hold on
277 x = (0:15:50);
    w = (A*cos(Omega12*x));
279 stem(x, w, 'y')
    hold off
281 %
    ylabel('\Omega = (2*pi)/16')
283 title('Sinal senoidal: A*cos( \Omega n)')

285 subplot(4,1,2)
    stem(n, (A*cos(Omega23*n)) )
287 %
    hold on
289 x = (0:(length(y)-1));
    stem(x, y, 'r')
291 hold off
    %
293 ylabel('\Omega = (2*pi)/(16/7)')

295 subplot(4,1,3)
    stem(n, (A*cos(Omega32*n)) )
297 %
    hold on
299 x = (0:(length(z)-1));
    stem(x, z, 'g')
301 hold off
    %
303 ylabel('\Omega = (2*pi)/(16/9)')

305 subplot(4,1,4)
    stem(n, (A*cos(Omega43*n)) )
307 %
    hold on
309 x = (0:(length(w)-1));
    stem(x, w, 'y')
311 hold off
    %
313 ylabel('\Omega = (2*pi)/(16/15)')

315
    %
317 % EOF
    %
```

Código 4.4: Demonstração da periodicidade de cosseno discreto.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: cos_periodicidade_discreto.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Titulo: Demo de
%         periodicidade de cos(.)
%         discreto
%
% Autor : Alexandre Santos de la Vega
% Data  : 22/10/2024
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% limpeza do ambiente de trabalho
%  variaveis
clear all
%  janelas
close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% - Periodicidade: (Omega * Np) = (K * 2*pi)
%
%   Período: Np = K * (2*pi / Omega)
%
%
% - Para: Omega = 2*pi / N  = 2*pi / (p/q) = (2*pi * q) / p
%   tem-se: Np = K * (p/q)
%
%     + Para: p e q inteiros
%       tem-se:
%         * Período fundamental: Nf = p
%         * Quantidade de (2*pi) ciclos para Nf: #Ciclos = q
%

%
p1 = 22;
q1 = 1;
N1 = (p1/q1);
%
p2 = 11;
q2 = 3;
N2 = (p2/q2);
%
p3 = 3*pi;
q3 = 1;
N3 = (p3/q3);
```

```
58
%
60 Omega1 = (2*pi)/N1;
Omega2 = (2*pi)/N2;
62 Omega3 = (2*pi)/N3;

64 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

66 %
n = 0:80;
68
%
70 x1 = cos(Omega1 * n);
x2 = cos(Omega2 * n);
72 x3 = cos(Omega3 * n);

74 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

76 %%%%%%%%%%

78

%
80 FigNbr = 0;
%
82 Omega1_deg = (Omega1 * 180 / pi);
Omega2_deg = (Omega2 * 180 / pi);
84 Omega3_deg = (Omega3 * 180 / pi);

86 %%%%%%%%%%

88 %
FigNbr = (FigNbr +1);
90 %
%
92 subplot(3,1,1)
figure(FigNbr)
94 stem(n,x1)
ylabel('x_1 [n]')
96 %
tit_str = {
98  ['Condição de periodicidade: (\Omega * N_p) = (K * 2 \pi)',...
    '  ;  ',...
100   'Período: N_p = K * (2 \pi / \Omega) .'],
  ['Para:  ',...
102   '\Omega = 2 \pi / N  = 2 \pi / (p/q) = (2 \pi * q) / p ,  ',...
    'tem-se:  ',...
104   'N_p = K * (p/q) ,  N_f = p  e  Quant\_2\pi\_ciclos\_para\_N_f = q . '],
  ['Sinal periódico:  x_1 [n] = cos(\Omega_1 n)',...
106   '  ;  ',...
    ' \Omega_1 = 2 \pi / N_1 = ', num2str(Omega1), ' rad = ',...
108   num2str(Omega1_deg), ' graus',...
    '  ;  ',...
110   'N_1 = p_1/q_1 = ', num2str(p1), '/', num2str(q1), ' .'],
};
112 title(tit_str)
%
114 %
subplot(3,1,2)
116 figure(FigNbr)
```

```
stem(n,x2)
ylabel('x_2 [n]')
%
tit_str = {
 ['Sinal periódico:  x_2 [n] = cos(\Omega_2 n)',...
  '   ;   ',...
  '\Omega_2 = 2 \pi / N_2 = ', num2str(Omega2), ' rad = ',...
  num2str(Omega2_deg), ' graus',...
  '   ;   ',...
  'N_2 = p_2/q_2 = ', num2str(p2), '/', num2str(q2), ' .']
};
title(tit_str)
%
%
subplot(3,1,3)
figure(FigNbr)
stem(n,x3)
ylabel('x_3 [n]')
%
tit_str = {
 ['Sinal NÃO periódico:  x_3 [n] = cos(\Omega_3 n)',...
  '   ;   ',...
  '\Omega_3 = 2 \pi / N_3 = ', num2str(Omega3), ' rad = ',...
  num2str(Omega3_deg), ' graus',...
  '   ;   ',...
  'N_3 = p_3/q_3 = ', num2str(p3), '/', num2str(q3), ' .']
};
title(tit_str)
%
xlabel('n')

%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% EOF
%
```

Código 4.5: Demonstração da periodicidade de cosseno amostrado.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: cos_periodicidade_amostrado.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Titulo: Demo de
%         periodicidade de cos(.)
%         amostrado
%
% Autor : Alexandre Santos de la Vega
% Data  : 30/03/2k9 ; 03/09/2k9
```

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

% limpeza do ambiente de trabalho
% variaveis
clear all
% janelas
close all


% definicao dos parametros
ne = 0:1000;
Femu = 500;
Temu = 1/Femu;


n = 0:30;

Fs1 = 24;
Ts1 = 1/Fs1;

Fs2 = 25;
Ts2 = 1/Fs2;

Fs3 = 10*sqrt(3);
Ts3 = 1/Fs3;

f1 = 3;
f2 = 3;
f3 = 3;


% calculo das sequencias
x1a = cos (2*pi*f1*(ne*Temu));
x2a = cos (2*pi*f2*(ne*Temu));
x3a = cos (2*pi*f3*(ne*Temu));

x1 = cos (2*pi*f1*(n*Ts1));
x2 = cos (2*pi*f2*(n*Ts2));
x3 = cos (2*pi*f3*(n*Ts3));


% graficos
%
% obtem tamanho de fullscreen
scrsz = get(0,'ScreenSize');  %[left,bottom,width,height]
%
% tenta evitar top bar e bottom bar
% ajusta tipo de papel
h1 = figure('PaperType','a4', ...
    'Position', [scrsz(1) (0.07 * scrsz(4)) scrsz(3) (0.81 * scrsz(4))]);
%
% graficos amostrados
subplot(3,2,2)
stem(n, x1)
%
hold on
r = 0:8;
```

```
76  stem(r, x1(r+1), 'r')
    hold off
78  %
    title('Sinal amostrado: cos (\Omega n) = cos(2 \pi f n Ts)')
80  ylabel('\Omega_1 = 1/4 \pi ; N_0 = 8')

82  subplot(3,2,4)
    stem(n, x2)
84  %
    hold on
86  r = 0:25;
    stem(r, x2(r+1), 'r')
88  hold off
    %
90  title('\Omega = (\omega Ts) = (2 \pi f Ts)')
    ylabel('\Omega_2 = 6/25 \pi ; N_0 = 25')
92
    subplot(3,2,6)
94  stem(n, x3)
    %
96  hold on
    r = 0:30;
98  stem(r, x3(r+1), 'r')
    hold off
100 %
    title('N_f = K_f * (F_S / f_0)')
102 ylabel('\Omega_3 = 6/(10 sqrt(3)) \pi ; Não periódico!')

104 xlabel('Amostra (n)')

106

    % graficos analogicos
108 subplot(3,2,1)
    plot((ne*Temu), x1a)
110 %
    hold on
112 x = (0:8)*Ts1;
    y = x1(1:length(x));
114 stem(x, y, 'r')
    hold off
116 %
    ylabel('F_{S_1} = 24 Hz')
118 title('Sinal analógico: cos(2 \pi f t) ; f_0 = 3 Hz')
    grid on
120
    subplot(3,2,3)
122 plot((ne*Temu), x2a)
    %
124 hold on
    x = (0:25)*Ts2;
126 y = x2(1:length(x));
    stem(x, y, 'r')
128 hold off
    %
130 ylabel('F_{S_2} = 25 Hz')
    grid on
132
    subplot(3,2,5)
134 plot((ne*Temu), x3a)
```

```
%
hold on
x = (0:30)*Ts3;
y = x3(1:length(x));
stem(x, y, 'r')
hold off
%
ylabel('F_{S_3} = 10 sqrt(3)  Hz')
xlabel('Tempo (s)')

grid on


%
% EOF
%
```

4.3 Efeitos das características relevantes de exponenciais

- O Código 4.6 apresenta a demonstração do processo de amostragem.
- O Código 4.7 apresenta a demonstração de ambiguidade em amplitude e fase.
- O Código 4.8 apresenta a demonstração de *aliasing* - 1.
- O Código 4.9 apresenta a demonstração de *aliasing* - 2.
- O Código 4.10 apresenta a demonstração de *aliasing*, interagindo com o usuário.
- O Código 4.11 apresenta a demonstração de cadeia de processamento com *aliasing*.

Código 4.6: Demonstração do processo de amostragem.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: amostragem.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               %
% Demos para                    %
% Apostila de DSP               %
%                               %
% Topico: processo de amostragem %
%                               %
% Autor : Alexandre Santos de la Vega %
% Modifs: /2010-01/2011-02/     %
%                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


% limpa ambiente
clear all
close all


% define parametros
A0  = 1;
F0  = 1e3;
T0  = 1/F0;
%
NOP  = 2;     % numero de periodos visualizados
%
PPPa = 1000;  % numero de pontos por periodo para sinal analogico
Tsa  = T0/PPPa;
t    = 0:Tsa:(NOP*T0);
%
PPPs = 10;    % numero de pontos por periodo para sinal amostrado
Tss  = T0/PPPs;
nTs  = 0:Tss:(NOP*T0);
%
```

```
n    = 0:(length(nTs) - 1);

% constroi sinais
xa = A0*cos(2*pi*F0*t);     % sinal analogico
xs = A0*cos(2*pi*F0*nTs);   % sinal amostrado

% quantiza sinais usando funcao definida externamente
NOI = 5;  % numero de intervalos do grid
%xaq = quant_half_grid(xa,-A0,(A0/NOI),A0);  % sinal analogico quantizado
xsq = quant_half_grid(xs,-A0,(A0/NOI),A0);  % sinal amostrado quantizado

% desenha graficos
figure(1)
%
subplot(3,2,1)
plot(t,xa)
ylabel('x(t)')
xlabel('t (s)')
title('Sinal analógico: x(t)')
%
subplot(3,2,3)
for k = 1:length(nTs)
    if (xs(k) == 0)
        stem(nTs(k),xs(k),'k')
    elseif (xs(k) > 0)
        stem(nTs(k),xs(k),'k','^')
    else
        stem(nTs(k),xs(k),'k','v')
    end
    hold on
end
ylabel('x(t) * \delta_{T_S}(t)')
xlabel('t (s)')
title('Sinal multiplicado pelo trem de impulsos: x(t) * \delta_{T_S}(t)')
%
subplot(3,2,4)
stem(nTs,xs)
ylabel('x[n T_s]')
xlabel('t (s)')
title('Sinal amostrado: x[n T_s] = x(t), para t = n T_s')
%
subplot(3,2,2)
stem(nTs,ones(1,length(nTs)),'k','^')
ylabel('\delta_{T_S}(t)')
xlabel('t (s)')
title('Trem de impulsos: \delta_{T_S}(t)')
%
subplot(3,2,5)
%stem(nTs,[xs ; xsq]')  % 'funciona, mas as cores nao ficam ok! ...'
stem(nTs,xs,'bo-')
hold on
stem(nTs,xsq,'ro-')
ylabel('[x[n T_S]]_Q')
xlabel('t (s)')
title('Sinal digital: [x[n T_S]]_Q = [x(t)]_Q, para t = n T_S')
%
```

```
102 subplot(3,2,6)
    %stem(n,[xs ; xsq]')  % 'funciona, mas as cores nao ficam ok! ...'
104 stem(n,xs,'bo-')
    hold on
106 stem(n,xsq,'ro-')
    ylabel('[ x[n] ]_Q')
108 xlabel('n')
    title('Sequência digital: [ x[n] ]_Q = [x[n T_S]]_Q')
110

112 %
    % EOF
114 %
```

Código 4.7: Demonstração de ambiguidade em amplitude e fase.

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2   %
    % Arquivo: ambiguidade_amp_fase_omega_pi.m
4   %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6

8   %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
10  %
    % Titulo: Demo de
12  %          superposicao de espectro
    %          cos(pi*n + Theta0) = cos(pi * n)
14  %
    % Autor : Alexandre Santos de la Vega
16  % Data  : / 2010-02 / 2011-01 /
    %
18  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
20
    % limpeza do ambiente de trabalho
22  %  variaveis
    clear all
24  %  janelas
    close all
26

28  % definicao dos parametros
    ne = 0:500;
30  Femu = 500;
    Temu = 1/Femu;
32
    n = 0:15;
34  %
    F0 = 3;
36  Theta0 = (2*pi)/8;
    %
38  Fs = 2*F0;
    Ts = 1/Fs;
40
    % calculo das sequencias
```

```
x1a = cos ( (2*pi*F0*(ne*Temu)) + Theta0 );
x2a = cos ( (2*pi*F0*(ne*Temu))          ) * cos(Theta0);

x1  = cos ( (2*pi*F0*(n*Ts))    + Theta0 );
x2  = cos ( (2*pi*F0*(n*Ts))             ) * cos(Theta0);


% graficos
%
figure(1)
set(gcf,"papertype","a4")
%
NORS = 6;
r = 0:NORS;
%
% graficos amostrados
subplot(2,2,2)
stem(n, x1)
%
hold on
stem(r, x1(r+1), 'r')
hold off
%
title('Sinais amostrados: x_1[n] = x_2[n]')
ylabel('x_1[n] = cos(\pi n + \Theta_0)')
AV = axis;
axis([AV(1) AV(2) (-1) (1)])

subplot(2,2,4)
stem(n, x2)
%
hold on
stem(r, x2(r+1), 'r')
hold off
%
title('\Omega = (\omega Ts) = (2 \pi f Ts) = \pi rad')
ylabel('x_2[n] = cos(\pi n) * cos(\Theta_0)')
xlabel('Amostra (n)')
AV = axis;
axis([AV(1) AV(2) (-1) (1)])


% graficos analogicos
%
x = (r)*Ts;
%
subplot(2,2,1)
plot((ne*Temu), x1a)
%
hold on
y = x1(1:length(x));
stem(x, y, 'r')
hold off
%
title('Sinais analógicos: x_1(t) \neq x_2(t)')
ylabel('x_1(t) = cos(2 \pi f_0 t + \Theta_0)')
%
grid on

```

```
subplot(2,2,3)
plot((ne*Temu), x2a)
%
hold on
y = x2(1:length(x));
stem(x, y, 'r')
hold off
%
title('f_0 = 3 Hz  ;  F_S = (2 f_0) Hz  ;  \Theta_0 = (2 \pi) / 8 rad')
ylabel('x_2(t) = cos(2 \pi f_0 t) * cos(\Theta_0)')
xlabel('Tempo (s)')
AV = axis;
axis([AV(1) AV(2) (-1) (1)])
%
grid on

%

figure(2)
set(gcf,"papertype","a4")
%
x = (r)*Ts;
%
y = x1(1:length(x));
stem(x, y, 'r')
hold on
plot((ne*Temu), x1a, 'k')
plot((ne*Temu), x2a, 'b')
%
tit_str_1 = ['Sinais analógicos: ', ...
             'x_1(t) = cos(2 \pi f_0 t + \Theta_0) ', ...
             '\neq cos(2 \pi f_0 t) * cos(\Theta_0) = x_2(t)'];
tit_str_2 = ['f_0 = 3 Hz  ;  ', ...
             'F_S = (2 f_0) Hz  ;  ', ...
             '\Theta_0 = (2 \pi) / 8 rad  ;  ', ...
             '\Omega = \pi rad'];
tit_str_3 = ['Sinais amostrados: ', ...
             'x_1[n] = cos (\pi n + \Theta_0) ', ...
             '= cos(\pi n) * cos(\Theta_0) = x_2[n]'];
title({tit_str_1,tit_str_2,tit_str_3})
ylabel('x_1(t) \neq x(_2(t)  ;  x_1[n] = x_2[n]')
xlabel('Tempo (s)')
%
%grid on


%
% EOF
%
```

Código 4.8: Demonstração de *aliasing* - 1.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: aliasing_demo1.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Titulo: Demo de superposicao de espectro
%
% Autor : Alexandre Santos de la Vega
% Data  : 30/03/2k9 ; 03/09/2k9
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

% limpeza do ambiente de trabalho
%  variaveis
clear all
%  janelas
close all


% definicao dos parametros
ne = 0:400;
%
Femu = 500;
Temu = 1/Femu;

n = 0:20;
%
f1 = 3;
f2 = 7;
f3 = 13;
%
Fs = 10;
Ts = 1/Fs;


% calculo das sequencias
x1a = cos (2*pi*f1*(ne*Temu));
x2a = cos (2*pi*f2*(ne*Temu));
x3a = cos (2*pi*f3*(ne*Temu));

x1 = cos (2*pi*f1*(n*Ts));
x2 = cos (2*pi*f2*(n*Ts));
x3 = cos (2*pi*f3*(n*Ts));


% graficos
%
figure(1)
set(gcf,"papertype","a4")
%
NORS = 5;
r = 0:NORS;
%
% graficos amostrados
subplot(3,2,2)
stem(n, x1)
hold on
stem(r, x1(r+1), 'r')
```

```
65 hold off
   %
67 ylabel('\Omega_1 = (0.6 \pi) rad')
   title('Sinal amostrado:  cos (\Omega n) = cos(2 \pi f n Ts)')
69
   subplot(3,2,4)
71 stem(n, x2)
   %
73 hold on
   stem(r, x2(r+1), 'r')
75 hold off
   %
77 ylabel('\Omega_2 = (2 \pi) - (0.6 \pi) rad')
   title('\Omega = (\omega Ts) = (2 \pi f Ts)')
79
   subplot(3,2,6)
81 stem(n, x3)
   %
83 hold on
   stem(r, x3(r+1), 'r')
85 hold off
   %
87 ylabel('\Omega_3 = (2 \pi) + (0.6 \pi) rad')
   title('F_S = 10 Hz')
89
   xlabel('Amostra (n)')
91

93 % graficos analogicos
   x = (r)*Ts;
95 %
   subplot(3,2,1)
97 plot((ne*Temu), x1a)
   %
99 hold on
   y = x1(1:length(x));
101 stem(x, y, 'r')
   hold off
103 %
   ylabel('f_1 = 3 Hz')
105 title('Sinal analógico: cos(2 \pi f t)')
   grid on
107
   subplot(3,2,3)
109 plot((ne*Temu), x2a)
   %
111 hold on
   x = (0:5)*Ts;
113 y = x2(1:length(x));
   stem(x, y, 'r')
115 hold off
   %
117 ylabel('f_2 = 7 Hz')
   grid on
119
   subplot(3,2,5)
121 plot((ne*Temu), x3a)
   %
123 hold on
```

```
x = (0:5)*Ts;
y = x3(1:length(x));
stem(x, y, 'r')
hold off
%
ylabel('f_3 = 13 Hz')
xlabel('Tempo (s)')
%
grid on

%

figure(2)
set(gcf,"papertype","a4")
%
x = (r)*Ts;
%
y = x1(1:length(x));
stem(x, y, 'r')
hold on
plot((ne*Temu), x1a, ':r')
plot((ne*Temu), x2a, '-.b')
plot((ne*Temu), x3a, 'k')
%
title({'Sinal analógico: cos(2 \pi f t)' , ...
       'Sinal amostrado:  cos (\Omega n) = cos(2 \pi f n Ts)'})
ylabel('f_1 = 3 Hz  ;  f_2 = 7 Hz  ;  f_3 = 13 Hz  ;  F_S = 10 Hz')
xlabel('Tempo (s)')
%
%grid on

%
% EOF
%
```

Código 4.9: Demonstração de *aliasing* - 2.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: aliasing_demo2.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Titulo: Demo de superposicao de espectro
%
% Autor : Alexandre Santos de la Vega
% Data  : 30/03/2k9 ; 03/09/2k9
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


% limpeza do ambiente de trabalho
%  variaveis
```

```
21 clear all
   % janelas
23 close all

25
   % definicao dos parametros
27 ne = 0:400;
   %
29 Femu = 500;
   Temu = 1/Femu;
31

33 n = 0:20;

35 Fs1 = 100;
   Ts1 = 1/Fs1;
37
   Fs2 = 10;
39 Ts2 = 1/Fs2;

41 Fs3 = 3/0.7;
   Ts3 = 1/Fs3;
43
   f1 = 3;
45 f2 = 3;
   f3 = 3;
47

49 % calculo das sequencias
   x1a = cos (2*pi*f1*(ne*Temu));
51 x2a = cos (2*pi*f2*(ne*Temu));
   x3a = cos (2*pi*f3*(ne*Temu));
53
   x1 = cos (2*pi*f1*(n*Ts1));
55 x2 = cos (2*pi*f2*(n*Ts2));
   x3 = cos (2*pi*f3*(n*Ts3));
57

59 % graficos

61 figure(1)
   set(gcf,"papertype","a4")
63 %
   %
65 % graficos amostrados
   subplot(3,2,2)
67 stem(n, x1)
   %
69 hold on
   r = 0:11;
71 stem(r, x1(r+1), 'r')
   hold off
73 %
   ylabel('\Omega_1 = (0.06 \pi) rad')
75 title('Sinal amostrado:  cos (\Omega n) = cos(2 \pi f n Ts)')

77 subplot(3,2,4)
   stem(n, x2)
79 %
```

```
hold on
r = 0:7;
stem(r, x2(r+1), 'r')
hold off
%
ylabel('\Omega_2 = (0.6 \pi) rad')
title('\Omega = (\omega Ts) = (2 \pi f Ts)')

subplot(3,2,6)
stem(n, x3)
%
hold on
r = 0:3;
stem(r, x3(r+1), 'r')
hold off
%
ylabel('\Omega_3 = 1.4 \pi = (2 \pi) - (0.6 \pi) rad')
xlabel('Amostra (n)')
%
%
% graficos analogicos
subplot(3,2,1)
plot((ne*Temu), x1a)
%
hold on
x = (0:11)*Ts1;
y = x1(1:length(x));
stem(x, y, 'r')
hold off
%
ylabel('F_{S_1} = 100 Hz')
title('Sinal analógico: cos(2 \pi f t) ; f_0 = 3 Hz')
grid on

subplot(3,2,3)
plot((ne*Temu), x2a)
%
hold on
x = (0:7)*Ts2;
y = x2(1:length(x));
stem(x, y, 'r')
hold off
%
ylabel('F_{S_2} = 10 Hz')
grid on

subplot(3,2,5)
plot((ne*Temu), x3a)
%
hold on
x = (0:3)*Ts3;
y = x3(1:length(x));
stem(x, y, 'r')
hold off
%
ylabel('F_{S_3} = (3 / 0.7) Hz')
xlabel('Tempo (s)')
%
grid on
```

```

%
% EOF
%
```

Código 4.10: Demonstração de *aliasing*, interagindo com o usuário.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: aliasing_ask_user.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                           %
% Titulo:                                                   %
% Demonstração de aliasing, interagindo com o usuário.     %
%                                                           %
% Autor : Alexandre Santos de la Vega                       %
% Data  : 01/09/2k8                                         %
%                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


%
clear all
close all

%
F1 =  250;
F2 = 1250;
F3 = 2250;


%
Tmin = 1/F3;
Tmax = 1/F1;

%
NbrPtsPer = 100;

%
TStepAna = Tmin/NbrPtsPer;
ta = 0:TStepAna:2*Tmax;

%
xa1 = cos(2*pi*F1*ta);
xa2 = cos(2*pi*F2*ta);
xa3 = cos(2*pi*F3*ta);


%
disp(' ')
disp(['Componentes do sinal com:  ', ...
```

```
        'f_1 = 250 Hz, f_2 = 1250 Hz e f_3 = 2250 Hz.'])
52  disp(' ')
    Fs = input('Frequência de amostragem (Hz):  ');
54  Ts = 1/Fs;
    nTs = 0:Ts:2*Tmax;
56
    %
58  xd1 = cos(2*pi*F1*nTs);
    xd2 = cos(2*pi*F2*nTs);
60  xd3 = cos(2*pi*F3*nTs);

62
    %
64  FigCntr = 0;
    n=0:(length(nTs)-1);
66
    %
68  FigCntr = (FigCntr + 1);
    figure(FigCntr)
70  set(gcf,"papertype","a4")

72  subplot(3,2,1)
    plot(ta,xa1)
74  ylabel('f_1 = 250 Hz')
    title('x(t) = cos (2 \pi f t)')
76  subplot(3,2,3)
    plot(ta,xa2)
78  ylabel('f_2 = 1250 Hz')
    subplot(3,2,5)
80  plot(ta,xa3)
    ylabel('f_3 = 2250 Hz')
82  xlabel('t (s)')

84  subplot(3,2,2)
    stem(nTs,xd1)
86  title('x[n T_s] = cos (2 \pi f n T_s)')
    subplot(3,2,4)
88  stem(nTs,xd2)
    subplot(3,2,6)
90  stem(nTs,xd3)
    xlabel('n T_s (s)')
92

94  %
    FigCntr = (FigCntr + 1);
96  figure(FigCntr)
    set(gcf,"papertype","a4")
98
    subplot(3,2,1)
100 stem(nTs,xd1)
    title('x[n T_s] = cos (2 \pi f n T_s)')
102 subplot(3,2,3)
    stem(nTs,xd2)
104 subplot(3,2,5)
    stem(nTs,xd3)
106 xlabel('n T_s (s)')

108 subplot(3,2,2)
    stem(n,xd1)
```

```
title('x[n] = cos (\Omega n) ; \Omega = (2 \pi f T_s)')
subplot(3,2,4)
stem(n,xd2)
subplot(3,2,6)
stem(n,xd3)
xlabel('n')

%
FigCntr = (FigCntr + 1);
figure(FigCntr)
set(gcf,"papertype","a4")

subplot(3,2,1)
plot(ta,xa1)
ylabel('f_1 = 250 Hz')
title('x(t) = cos (2 \pi f t)')
subplot(3,2,3)
plot(ta,xa2)
ylabel('f_2 = 1250 Hz')
subplot(3,2,5)
plot(ta,xa3)
ylabel('f_3 = 2250 Hz')
xlabel('t (s)')

subplot(3,2,2)
stem(n,xd1)
title('x[n] = cos (\Omega n) ; \Omega = (2 \pi f T_s)')
subplot(3,2,4)
stem(n,xd2)
subplot(3,2,6)
stem(n,xd3)
xlabel('n')

%
FigCntr = (FigCntr + 1);
figure(FigCntr)
set(gcf,"papertype","a4")

subplot(3,3,1)
plot(ta,xa1)
ylabel('f_1 = 250 Hz')
title('x(t) = cos (2 \pi f t)')
subplot(3,3,4)
plot(ta,xa2)
ylabel('f_2 = 1250 Hz')
subplot(3,3,7)
plot(ta,xa3)
ylabel('f_3 = 2250 Hz')
xlabel('t (s)')

subplot(3,3,2)
stem(nTs,xd1)
title('x[n T_s] = cos (2 \pi f n T_s)')
subplot(3,3,5)
stem(nTs,xd2)
subplot(3,3,8)
stem(nTs,xd3)
```

```
xlabel('n T_s (s)')

n=0:(length(nTs)-1);

subplot(3,3,3)
stem(n,xd1)
title('x[n] = cos (\Omega n) ; \Omega = (2 \pi f T_s)')
subplot(3,3,6)
stem(n,xd2)
subplot(3,3,9)
stem(n,xd3)
xlabel('n')

%
% EOF
%
```

Código 4.11: Demonstração de cadeia de processamento com *aliasing*.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: DSP_2018_1_TS1_Gabarito.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Título:                                                                     %
% Demonstração de cadeia de processamento com aliasing                        %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Disciplina:                                                                 %
%   Processamento Digital de Sinais.                                          %
% Professor :                                                                 %
%   Alexandre Santos de la Vega.                                              %
% Período   :                                                                 %
%   2018-1.                                                                   %
%                                                                             %
% Atividade :                                                                 %
%   Teste Surpresa 1 - 2018_1 - Gabarito.                                     %
%                                                                             %
% Assunto   :                                                                 %
%   Cadeia de processamento com amostragem inadequada,                        %
%     causando aliasing, formada pelas seguintes etapas:                      %
%                                                                             %
%   Mixagem Analogica --> Amostragem --> Tx --> ...                           %
%     ... --> Rx --> Interpolacao --> Filtragem.                              %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%% %%%%%%%%%% %%%%%%%%%%

% Limpa ambiente

```

```
clear all
close all

%%%%%%%%%% %%%%%%%%%% %%%%%%%%%%

%%%%%%%%%% %%%%%%%%%% %%%%%%%%%%
% Parametros gerais
% Discretos
%
Fs = 44e3;
%
nd = 0:43;
nTs = nd/Fs;

% Analogicos
%
Fsa = 10*Fs;
%
n = 0:439;
t = n/Fsa;

%%%%%%%%%% %%%%%%%%%% %%%%%%%%%%

%%%%%%%%%% %%%%%%%%%% %%%%%%%%%%

% Sinais Tx

% Parametros

%
A1 = 1;
A2 = 1;
A3 = 1;
A4 = 1;

%
f1 =  2e3;
f2 = 21e3;
f3 = 24e3;
f4 = 40e3;


% Sinais analogicos

%
x1 = A1*cos(2*pi*f1*n/Fsa);
x2 = A2*cos(2*pi*f2*n/Fsa);
x3 = A3*cos(2*pi*f3*n/Fsa);
x4 = A4*cos(2*pi*f4*n/Fsa);

%
xlow  = x1;
xmed  = x2 + x3;
xhigh = x4;
```

```
98
    %
100 x = xlow + xmed + xhigh;

102
    % Sinais discretos Tx
104
    %
106 x1d = A1*cos(2*pi*f1*nd/Fs);
    x2d = A2*cos(2*pi*f2*nd/Fs);
108 x3d = A3*cos(2*pi*f3*nd/Fs);
    x4d = A4*cos(2*pi*f4*nd/Fs);
110
    %
112 xlowd  = x1d;
    xmedd  = x2d + x3d;
114 xhighd = x4d;

116 %
    xd = xlowd + xmedd + xhighd;
118
    %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%
120

122 %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%

124 % Sinais Rx

126
    % Parametros
128
    %
130 A5 = 1;
    A6 = 1;
132
    %
134 f5 =  4e3;
    f6 = 20e3;
136

138 % Sinais analogicos

140 %
    y1 = A1*cos(2*pi*f1*n/Fsa);
142 y2 = A2*cos(2*pi*f2*n/Fsa);
    y5 = A5*cos(2*pi*f5*n/Fsa);
144 y6 = A6*cos(2*pi*f6*n/Fsa);
    y7 = 0*n;
146
    %
148 ylow  = y1 + y5;
    ymed  = y2 + y6;
150 yhigh = y7;

152 %
    y = ylow + ymed + yhigh;
154
    %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%
156
```

```
158 %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%

160 % Graficos

162
    %%%%%%%%%%
164
    %
166 % Nota:
    %
168 %  Foram inseridos comandos de "return" para teste...
    %
170
    %%%%%%%%%%
172
    % Controle da numeracao dos graficos
174
    %
176 FigNbr = 0;

178 %%%%%%%%%%

180
    % Limites de abscissas
182
    %
184 min_t = min(t);
    max_t = max(t);
186 %
    min_nTs = min(nTs);
188 max_nTs = max(nTs);
    %
190 min_n = min(nd);
    max_n = max(nd);
192

194 % Limites de ordenadas

196 %
    min_amp = min(x);
198 max_amp = max(x);

200 %%%%%%%%%%

202 %
    FigNbr = FigNbr+1;
204 figure(FigNbr);
    set(gcf,"papertype","a4")
206 %
    subplot(4,3,1)
208 plot(t,x1)
    hold on
210 stem(nd/Fs,x1d)
    v = axis;
212 v(1) = min_t;
    v(2) = max_t;
214 %v(3) = min_amp;
    %v(4) = max_amp;
```

```
axis(v)
title('A_1 cos(2 \pi f_1 t)  ;  f_1 = 2 kHz')
ylabel('x_1 (t)')
%
subplot(4,3,4)
plot(t,x2)
hold on
stem(nd/Fs,x2d)
v = axis;
v(1) = min_t;
v(2) = max_t;
%v(3) = min_amp;
%v(4) = max_amp;
axis(v)
title('A_2 cos(2 \pi f_2 t)  ;  f_2 = 21 kHz')
ylabel('x_2 (t)')
%
subplot(4,3,7)
plot(t,x3)
hold on
stem(nd/Fs,x3d)
v = axis;
v(1) = min_t;
v(2) = max_t;
%v(3) = min_amp;
%v(4) = max_amp;
axis(v)
title('A_3 cos(2 \pi f_3 t)  ;  f_3 = 24 kHz')
ylabel('x_3 (t)')
%
subplot(4,3,10)
plot(t,x4)
hold on
stem(nd/Fs,x4d)
v = axis;
v(1) = min_t;
v(2) = max_t;
%v(3) = min_amp;
%v(4) = max_amp;
axis(v)
title('A_4 cos(2 \pi f_4 t)  ;  f_4 = 40 kHz')
ylabel('x_4 (t)')
%
xlabel('t')
%
subplot(4,3,2)
stem(nd/Fs,x1d,'r')
v = axis;
v(1) = min_t;
v(2) = max_t;
%v(3) = min_amp;
%v(4) = max_amp;
axis(v)
title('A_1 cos(2 \pi f_1 n T_S)  ;  F_S = 44 kHz')
ylabel('x_1 (n T_S)')
%
subplot(4,3,5)
stem(nd/Fs,x2d,'r')
v = axis;
```

```
v(1) = min_t;
v(2) = max_t;
%v(3) = min_amp;
%v(4) = max_amp;
axis(v)
title('A_2 cos(2 \pi f_2 n T_S)  ;  F_S = 44 kHz')
ylabel('x_2 (n T_S)')
%
subplot(4,3,8)
stem(nd/Fs,x3d,'r')
v = axis;
v(1) = min_t;
v(2) = max_t;
%v(3) = min_amp;
%v(4) = max_amp;
axis(v)
title('A_3 cos(2 \pi f_3 n T_S)  ;  F_S = 44 kHz')
ylabel('x_3 (n T_S)')
%
subplot(4,3,11)
stem(nd/Fs,x4d,'r')
v = axis;
v(1) = min_t;
v(2) = max_t;
%v(3) = min_amp;
%v(4) = max_amp;
axis(v)
title('A_4 cos(2 \pi f_4 n T_S)  ;  F_S = 44 kHz')
ylabel('x_4 (n T_S)')
%
xlabel('n T_S')
%
subplot(4,3,9)
plot(t,y6)
hold on
stem(nd/Fs,x3d)
v = axis;
v(1) = min_t;
v(2) = max_t;
%v(3) = min_amp;
%v(4) = max_amp;
axis(v)
title('A_6 cos(2 \pi f_6 t)  ;  f_6 = 20 kHz')
ylabel('y_6 (t)')
%
subplot(4,3,12)
plot(t,y5)
hold on
stem(nd/Fs,x4d)
v = axis;
v(1) = min_t;
v(2) = max_t;
%v(3) = min_amp;
%v(4) = max_amp;
axis(v)
title('A_5 cos(2 \pi f_5 t)  ;  f_5 = 4 kHz')
ylabel('y_5 (t)')
%
xlabel('t')
```

```
%
% Insercao nao convencional de texto...
%
%   Subplot vazio...
subplot(4,3,3)
axis("off")
%   Definicao de cell array com strings de comprimentos diferentes...
title_str = {'* Sinais analógicos em azul.' ,
             '' ,
             '* Sinais discretos em vermelho.' } ;
%   Insercao de texto no titulo de subplot vazio...
hdl = title(title_str ,
            "fontsize",14,
            "horizontalalignment","left",
            "verticalalignment","top");
title_position = get(hdl,"position");
title_position(1) = 0;
set(hdl,"position",title_position)
%
%return

%%%%%%%%%%

%
FigNbr = FigNbr+1;
figure(FigNbr);
set(gcf,"papertype","a4")
%
subplot(3,4,1)
plot(t,x1)
v = axis;
v(1) = min_t;
v(2) = max_t;
v(3) = min_amp;
v(4) = max_amp;
axis(v)
title('A_1 cos(2 \pi f_1 t)  ;  f_1 = 2 kHz')
ylabel('x_1 (t)')
%
xlabel('t')
%
subplot(3,4,2)
plot(t,x2)
v = axis;
v(1) = min_t;
v(2) = max_t;
v(3) = min_amp;
v(4) = max_amp;
axis(v)
title('A_2 cos(2 \pi f_2 t)  ;  f_2 = 21 kHz')
ylabel('x_2 (t)')
%
xlabel('t')
%
subplot(3,4,3)
plot(t,x3)
v = axis;
v(1) = min_t;
v(2) = max_t;
```

```
v(3) = min_amp;
v(4) = max_amp;
axis(v)
title('A_3 cos(2 \pi f_3 t)  ;  f_3 = 24 kHz')
ylabel('x_3 (t)')
%
xlabel('t')
%
subplot(3,4,4)
plot(t,x4)
v = axis;
v(1) = min_t;
v(2) = max_t;
v(3) = min_amp;
v(4) = max_amp;
axis(v)
title('A_4 cos(2 \pi f_4 t)  ;  f_4 = 40 kHz')
ylabel('x_4 (t)')
%
xlabel('t')
%
subplot(3,4,5)
stem(nd,x1d,'r')
v = axis;
v(1) = min_n;
v(2) = max_n;
v(3) = min_amp;
v(4) = max_amp;
axis(v)
title('x_1 [n]  ;  F_S = 44 kHz')
ylabel('x_1 [n]')
%
xlabel('n')
%
subplot(3,4,6)
stem(nd,x2d,'r')
v = axis;
v(1) = min_n;
v(2) = max_n;
v(3) = min_amp;
v(4) = max_amp;
axis(v)
title('x_2 [n]  ;  F_S = 44 kHz')
ylabel('x_2 [n]')
%
xlabel('n')
%
subplot(3,4,7)
stem(nd,x3d,'r')
v = axis;
v(1) = min_n;
v(2) = max_n;
v(3) = min_amp;
v(4) = max_amp;
axis(v)
title('x_3 [n] = y_6 [n]  ;  F_S = 44 kHz')
ylabel('x_3 [n]')
%
xlabel('n')
```

```
452 %
    subplot(3,4,8)
454 stem(nd,x4d,'r')
    v = axis;
456 v(1) = min_n;
    v(2) = max_n;
458 v(3) = min_amp;
    v(4) = max_amp;
460 axis(v)
    title('x_4 [n] = y_5 [n]   ;  F_S = 44 kHz')
462 ylabel('x_4 [n]')
    %
464 xlabel('n')
    %
466 subplot(3,4,11)
    plot(t,y6)
468 v = axis;
    v(1) = min_t;
470 v(2) = max_t;
    v(3) = min_amp;
472 v(4) = max_amp;
    axis(v)
474 title('A_6 cos(2 \pi f_6 t)   ;  f_6 = 20 kHz')
    ylabel('y_6 (t)')
476 %
    xlabel('t')
478 %
    subplot(3,4,12)
480 plot(t,y5)
    v = axis;
482 v(1) = min_t;
    v(2) = max_t;
484 v(3) = min_amp;
    v(4) = max_amp;
486 axis(v)
    title('A_5 cos(2 \pi f_5 t)   ;  f_5 = 4 kHz')
488 ylabel('y_5 (t)')
    %
490 xlabel('t')
    %
492 %return

494 %%%%%%%%%

496 %
    FigNbr = FigNbr+1;
498 figure(FigNbr);
    set(gcf,"papertype","a4")
500 %
    subplot(3,4,1)
502 plot(t,x1)
    v = axis;
504 v(1) = min_t;
    v(2) = max_t;
506 v(3) = min_amp;
    v(4) = max_amp;
508 axis(v)
    title('A_1 cos(2 \pi f_1 t)   ;  f_1 = 2 kHz')
510 ylabel('x_1 (t)')
```

```
%
subplot(3,4,9)
plot(t,xlow)
v = axis;
v(1) = min_t;
v(2) = max_t;
v(3) = min_amp;
v(4) = max_amp;
axis(v)
title('Sinal Tx (low)')
ylabel('x_{low} (t)')
%
xlabel('t')
%
subplot(3,4,2)
plot(t,x2)
v = axis;
v(1) = min_t;
v(2) = max_t;
v(3) = min_amp;
v(4) = max_amp;
axis(v)
title('A_2 cos(2 \pi f_2 t)  ;  f_2 = 21 kHz')
ylabel('x_2 (t)')
%
subplot(3,4,6)
plot(t,x3)
v = axis;
v(1) = min_t;
v(2) = max_t;
v(3) = min_amp;
v(4) = max_amp;
axis(v)
title('A_3 cos(2 \pi f_3 t)  ;  f_3 = 24 kHz')
ylabel('x_3 (t)')
%
subplot(3,4,10)
plot(t,xmed)
v = axis;
v(1) = min_t;
v(2) = max_t;
v(3) = min_amp;
v(4) = max_amp;
axis(v)
title('Sinal Tx (med)')
ylabel('x_{med} (t)')
%
xlabel('t')
%
subplot(3,4,3)
plot(t,x4)
v = axis;
v(1) = min_t;
v(2) = max_t;
v(3) = min_amp;
v(4) = max_amp;
axis(v)
title('A_4 cos(2 \pi f_4 t)  ;  f_4 = 40 kHz')
ylabel('x_4 (t)')
```

```
570 %
    subplot(3,4,11)
572 plot(t,xhigh)
    v = axis;
574 v(1) = min_t;
    v(2) = max_t;
576 v(3) = min_amp;
    v(4) = max_amp;
578 axis(v)
    title('Sinal Tx (high)')
580 ylabel('x_{high} (t)')
    %
582 xlabel('t')
    %
584 subplot(3,4,12)
    plot(t,x)
586 v = axis;
    v(1) = min_t;
588 v(2) = max_t;
    v(3) = min_amp;
590 v(4) = max_amp;
    axis(v)
592 title('Sinal Tx (low + med + high)')
    ylabel('x (t)')
594 %
    xlabel('t')
596 %
    %return
598
    %%%%%%%%%%
600
    %
602 FigNbr = FigNbr+1;
    figure(FigNbr);
604 set(gcf,"papertype","a4")
    %
606 subplot(4,1,1)
    plot(t,x,'b')
608 v = axis;
    v(1) = min_t;
610 v(2) = max_t;
    v(3) = min_amp;
612 v(4) = max_amp;
    axis(v)
614 title('Sinal Tx')
    ylabel('x (t)')
616 %
    subplot(4,1,2)
618 stem(nTs,xd,'r')
    v = axis;
620 v(1) = min_nTs;
    v(2) = max_nTs;
622 v(3) = min_amp;
    v(4) = max_amp;
624 axis(v)
    title('Sinal Discreto Tx')
626 ylabel('x (n T_s)')
    %
628 subplot(4,1,3)
```

```
    plot(t,y,'k')
630 v = axis;
    v(1) = min_t;
632 v(2) = max_t;
    v(3) = min_amp;
634 v(4) = max_amp;
    axis(v)
636 title('Sinal Rx')
    ylabel('y (t)')
638 %
    subplot(4,1,4)
640 plot(t,x,'b')
    hold on
642 stem(nTs,xd,'r')
    plot(t,y,'k')
644 v = axis;
    v(1) = min_t;
646 v(2) = max_t;
    v(3) = min_amp;
648 v(4) = max_amp;
    axis(v)
650 title('Superposição dos três sinais')
    %
652 xlabel('t')
    %
654 %return

656 %%%%%%%%%

658 %
    FigNbr = FigNbr+1;
660 figure(FigNbr);
    set(gcf,"papertype","a4")
662 %
    subplot(3,4,2)
664 plot(t,y1)
    v = axis;
666 v(1) = min_t;
    v(2) = max_t;
668 v(3) = min_amp;
    v(4) = max_amp;
670 axis(v)
    title('A_1 cos(2 \pi f_1 t)  ;  f_1 = 2 kHz')
672 ylabel('y_1 (t)')
    %
674 subplot(3,4,3)
    plot(t,y2)
676 v = axis;
    v(1) = min_t;
678 v(2) = max_t;
    v(3) = min_amp;
680 v(4) = max_amp;
    axis(v)
682 title('A_2 cos(2 \pi f_2 t)  ;  f_2 = 21 kHz')
    ylabel('y_2 (t)')
684 %
    subplot(3,4,10)
686 plot(t,ylow)
    v = axis;
```

```
688 v(1) = min_t;
    v(2) = max_t;
690 v(3) = min_amp;
    v(4) = max_amp;
692 axis(v)
    title('Sinal Rx (low)')
694 ylabel('y_{low} (t)')
    %
696 xlabel('t')
    %
698 subplot(3,4,6)
    plot(t,y5)
700 v = axis;
    v(1) = min_t;
702 v(2) = max_t;
    v(3) = min_amp;
704 v(4) = max_amp;
    axis(v)
706 title('A_5 cos(2 \pi f_5 t)  ;  f_5 = 4 kHz')
    ylabel('y_5 (t)')
708 %
    subplot(3,4,7)
710 plot(t,y6)
    v = axis;
712 v(1) = min_t;
    v(2) = max_t;
714 v(3) = min_amp;
    v(4) = max_amp;
716 axis(v)
    title('A_6 cos(2 \pi f_6 t)  ;  f_6 = 20 kHz')
718 ylabel('y_6 (t)')
    %
720 subplot(3,4,11)
    plot(t,ymed)
722 v = axis;
    v(1) = min_t;
724 v(2) = max_t;
    v(3) = min_amp;
726 v(4) = max_amp;
    axis(v)
728 title('Sinal Rx (med)')
    ylabel('y_{med} (t)')
730 %
    xlabel('t')
732 %
    subplot(3,4,4)
734 plot(t,y7)
    v = axis;
736 v(1) = min_t;
    v(2) = max_t;
738 v(3) = min_amp;
    v(4) = max_amp;
740 axis(v)
    title('Sinal nulo')
742 ylabel('y_7 (t)')
    %
744 subplot(3,4,12)
    plot(t,yhigh)
746 v = axis;
```

```
    v(1) = min_t;
748 v(2) = max_t;
    v(3) = min_amp;
750 v(4) = max_amp;
    axis(v)
752 title('Sinal Rx (high)')
    ylabel('y_{high} (t)')
754 %
    xlabel('t')
756 %
    subplot(3,4,9)
758 plot(t,y)
    v = axis;
760 v(1) = min_t;
    v(2) = max_t;
762 v(3) = min_amp;
    v(4) = max_amp;
764 axis(v)
    title('Sinal Rx (low + med + high)')
766 ylabel('y (t)')
    %
768 xlabel('t')
    %
770 %return

772 %%%%%%%%%%

774 %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%

776

    %
778 % EOF
    %
```

Parte III

Representações de um SLIT

Capítulo 5

Representações de um SLIT

5.1 Introdução

Diversas descrições para os sistemas do tipo SLIT, Sistema Linear Invariante ao Tempo (ou ao deslocamento), são abordadas na terceira parte do curso.

No domínio do tempo, são discutidas as seguintes representações: resposta ao impulso, equação de diferença, operador de transferência, equações de estado, diagrama de sistema (ou realização ou estrutura).

Nas seções que se seguem, são apresentadas diversas listagens de programas, relativas a tais representações.

5.2 Operador de transferência

5.2.1 Diagrama de Pólos e Zeros (DPZ)

- O Código 5.1 apresenta um exemplo de traçado de Diagrama de Pólos e Zeros (DPZ). No caso da ocorrência de singularidades múltiplas, as mesmas são sinalizadas com um número indicativo da multiplicidade.

Código 5.1: Traçado de DPZ, com sinalização de singularidades múltiplas.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: DPZ_indica_mult_sing.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                             %
% DSP   : Demo sobre tracado de DPZ,          %
%         com sinalizacao de singularidades multiplas. %
%                                             %
% Autor: Alexandre S. de la Vega.             %
%                                             %
% Datas: /2018-1/                             %
%                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
```

```
%%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%
%
% limpa workspace
clear all
close all

%
%%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%
%
% calcula circulo de raio unitario
N = 360;
n = 0:(N-1);
unit_circ = exp(j*2*pi*n/N);

%
%%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%
%
% define funcao polinomial racional
% para testes...

%   numerador
%
%num = [ 5 4 3 2 1 ];
%
num = [ 1 2 3 4 5 ];
%num = [ 1 2 3 4 ];
%num = [ 1 2 ];
%num = 1;
%
%num = [ 1 1 .25 ];
%
% erro numerico em roots()
% gera 2 zeros simples
% ao inves de 1 polo multiplo com m=2 ...
%num = [ 1.00000   2.50000   2.00000   0.50000 ];

%   denominador
%
den = [ 5 4 3 2 1 ];
%den = [ 5 4 3 2 ];
%den = [ 5 4 3 ];
%den = [ 5 4 ];
%den = 5;
%
%den = [ 1 -1 .25 ];
%

%
%%%%%%%%%%
%

nc = [ 1 .9 ];
np = [ 1 .8 ];
nf = [ 1 .5 ];

num = conv( [1 0] , conv(nc,np) );
```

```
79  den = [ 1 0 0 0 ] + conv( conv(nc,np) , nf );

81  %
    %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%
83  %

85  % calcula zeros, polos e constante de ganho
    z = roots(num);
87  p = roots(den);
    k = num(1)/den(1);
89
    %
91  %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%
    %
93
    % define zeros e polos
95  % para testes...
    %
97  % zeros multiplos
    %z = [ .2 .3 .1 .2 .3 .3 .3 .1 .2 ]';
99  %
    % polos multiplos
101 %p = [ -.3-.3*i -.3+.3*i , ...
    %       -.1-.1*i -.1+.1*i , ...
103 %       -.2-.2*i -.2+.2*i , ...
    %       -.2-.2*i -.2+.2*i , ...
105 %       -.3-.3*i -.3+.3*i , ...
    %       -.3-.3*i -.3+.3*i , ...
107 %       -.3-.3*i -.3+.3*i , ...
    %       -.1-.1*i -.1+.1*i , ...
109 %       -.2-.2*i -.2+.2*i , ...
    %       0]';
111
    %
113 %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%
    %
115
    % gerencia as singularidades simples e multiplas
117
    %   zeros
119 %
    %   seleciona os diferentes valores de zeros
121 z_unq = unique(z);

123 %   separa os zeros simples dos multiplos
    %   e
125 %   identifica as multiplicidades
    %
127 z_smp = [];
    z_mlt = [];
129 z_mlt_m = [];
    %
131 for ind = 1:length(z_unq)
      unq_pos = find( z == z_unq(ind) );
133   m = length(unq_pos);
      if ( m == 1)
135     z_smp = [ z_smp ; z_unq(ind) ];
      else
137     z_mlt   = [ z_mlt   ; z_unq(ind) ];
```

```
    z_mlt_m = [ z_mlt_m ; m           ];
  endif
endfor
%
clear  unq_pos  m

%   polos
%
%   seleciona os diferentes valores de polos
p_unq = unique(p);

%   separa os polos simples dos multiplos
%   e
%   identifica as multiplicidades
%
p_smp = [];
p_mlt = [];
p_mlt_m = [];
%
for ind = 1:length(p_unq)
  unq_pos = find( p == p_unq(ind) );
  m = length(unq_pos);
  if ( m == 1)
    p_smp = [ p_smp ; p_unq(ind) ];
  else
    p_mlt   = [ p_mlt   ; p_unq(ind) ];
    p_mlt_m = [ p_mlt_m ; m          ];
  endif
endfor
%
clear  unq_pos  m

%
%%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%
%


%
% cria grafico
%

% calculos comuns a todos os graficos
%
re_z_unq = real(z_unq);
im_z_unq = imag(z_unq);
%
re_p_unq = real(p_unq);
im_p_unq = imag(p_unq);
%
mult_offset_x = .025;
mult_offset_y = .050;

% calcula raio de visualizacao,
% baseado na posicao das singularidades (zeros e polos)
%
if ( isempty(z) && isempty(p) )
  dpz_limit = 1;
else
  max_re_sing = max( [max(abs(re_z_unq)) , max(abs(re_p_unq))] );
  max_im_sing = max( [max(abs(im_z_unq)) , max(abs(im_p_unq))] );
```

```
197   max_sing     = max( [max_re_sing     , max_im_sing    ] );
      dpz_limit = fix(max_sing) + 1;
199 endif

201 % cria canvas
    hf=figure(1);
203 set(hf,'papertype','a4')

205 % desenha circulo de raio unitario
    plot(real(unit_circ),imag(unit_circ),'b')
207
    % mantem todos os graficos na mesma figura
209 hold on

211 % desenha os zeros simples
    plot(real(z_smp),imag(z_smp),'ok')
213
    % desenha os zeros multiplos
215 plot(real(z_mlt),imag(z_mlt),'ok')
    for ind = 1:length(z_mlt_m)
217   text( (real(z_mlt(ind)) + mult_offset_x) ,
          (imag(z_mlt(ind)) + mult_offset_y) ,
219       num2str(z_mlt_m(ind))                 )
    endfor
221
    % desenha os polos simples
223 plot(real(p_smp),imag(p_smp),'xk')

225 % desenha os polos multiplos
    plot(real(p_mlt),imag(p_mlt),'xk')
227 for ind = 1:length(p_mlt_m)
      text( (real(p_mlt(ind)) + mult_offset_x) ,
229       (imag(p_mlt(ind)) + mult_offset_y) ,
          num2str(p_mlt_m(ind))                 )
231 endfor

233 % escreve constante de ganho
    % em posicao definida pelos limites do grafico
235 K_str = ['K = ', sprintf("%.3f",k)];
    K_str_offset = dpz_limit / 8;
237 text( ((-dpz_limit) + K_str_offset) ,
        (( dpz_limit) - K_str_offset) ,
239     K_str                          )

241 % identifica o grafico
    title('Diagrama de polos e zeros (DPZ)')
243 ylabel('Im\{.\}')
    xlabel('Re\{.\}')
245
    % habilita o grid na figura
247 grid on

249 % Controla formato do grafico
    % Deve vir ANTES do controle dos limites !!!
251 axis("image")

253 % controla limites
    v = [ (-dpz_limit) dpz_limit (-dpz_limit) dpz_limit ];
255 axis(v)
```

```
257 %
    %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%% %%%%%%%%%%
259 %

261 %
    % EOF
263 %
```

Parte IV

Sinais e sistemas no domínio da frequência

Capítulo 6

Sinais no domínio da frequência

6.1 Introdução

Na terceira parte do curso, são abordados os seguintes itens sobre sinais (com tempo discreto) no domínio da frequência: revisão das representações em frequência com tempo contínuo (Série de Fourier, Transformada de Fourier, Transformada de Laplace), Série de Fourier de Tempo Discreto (DTFS), Transformada de Fourier de Tempo Discreto (DTFT), Transformada de Fourier Discreta (DFT), Transformada Rápida de Fourier (FFT), Transformada Z, relações entre as diversas representações, parâmetros e efeitos importantes. Além disso, são também abordados os seguintes tópicos sobre SLIT (com tempo discreto) no domínio da frequência: Resposta em Frequência, Seletividade em Frequência, Função de Transferência ou Função de Sistema, representações de um SLIT no domínio da frequência. Nas seções que se seguem, são apresentadas diversas listagens de programas, relativas a tais assuntos.

6.2 DTFS

- O Código 6.1 apresenta o cálculo dos coeficientes da DTFS de um sinal discreto periódico.

Código 6.1: Exemplo de cálculo dos coeficientes da DTFS.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: dtfs_exm.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DTFS example                    %
%                                 %
% Author: Alexandre S. de la Vega %
% Version: /2022_01_12/           %
%          /2010_12_14/           %
%                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

%%%%%%%%% %%%%%%%%% %%%%%%%%%
```

```
% clear workspace
clear all
close all

%%%%%%%%%% %%%%%%%%%% %%%%%%%%%%

% define input sequence
%
x_fund_n = rand(1,10);
%
x_fund_n =  5:-1:-10;
x_fund_n = 10:-1:-5;
%
%x_fund_n = 10:-1:-10;
%x_fund_n = 10:-1:0;
%x_fund_n = 0:-1:-10;

%
N = length(x_fund_n);

%%%%%%%%%% %%%%%%%%%% %%%%%%%%%%

% calculate DTFS coeficients
X_fund_k = zeros(1,N);
for k = 0:(N-1)
  ind = k+1;
  for n = 0:(N-1)
    X_fund_k(ind) = X_fund_k(ind) + ...
                    ( x_fund_n(n+1) * exp(j*k*((2*pi)/N)*n) );
  end
end
X_fund_k = X_fund_k / N;
X_fund_k_mod = abs(X_fund_k);
X_fund_k_deg = (angle(X_fund_k)*180)/pi;

%%%%%%%%%% %%%%%%%%%% %%%%%%%%%%

%
% draw sequences
%

%%%%%%%%%% %%%%%%%%%%

%
FigNbr = 0;

%
bck_clr = [.1, .1, .1];

%
n = 0:(N-1);
k = 0:(N-1);

%
%abs_min = min(abs(x_fund_n));
abs_max = max(abs(x_fund_n));

%
%val_min = min(x_fund_n);
```

```
%val_max = max(x_fund_n);

%
x_per_n = [x_fund_n x_fund_n x_fund_n];
n_per   = 0:(length(x_per_n)-1);

%
X_per_k_mod = [X_fund_k_mod X_fund_k_mod X_fund_k_mod];
X_per_k_deg = [X_fund_k_deg X_fund_k_deg X_fund_k_deg];
k_per   = 0:(length(X_per_k_mod)-1);

%%%%%%%%%% %%%%%%%%%%

%
FigNbr = FigNbr+1;
figure(FigNbr)
%
%
subplot(2,4,1)
stem(n_per, x_per_n)
axis([n_per(1) n_per(end) -abs_max abs_max])
set(gca, "color", bck_clr)
ylabel('x_{per} [n]')
xlabel('n')
title({'Periodic signal', 'x_{per} [n]'})
%
%
subplot(2,4,2)
stem(n, x_fund_n)
axis([n(1) n(end) -abs_max abs_max])
set(gca, "color", bck_clr)
ylabel('x_{fund} [n]')
xlabel('n')
title({'Representation of a periodic signal',
       'Time (n)'})
%
%
subplot(2,4,3)
stem(k, X_fund_k_mod)
axis([k(1) k(end) 0 abs_max])
set(gca, "color", bck_clr)
ylabel('| X_{fund} [k] |')
title({'Representation of a periodic signal',
       'Frequency (k)'})
%
subplot(2,4,7)
stem(k, X_fund_k_deg)
axis([k(1) k(end) -200 200 ])
set(gca, "color", bck_clr)
ylabel('\angle X_{fund} [k]   (degree)')
xlabel('k')
%
%
subplot(2,4,4)
stem(k_per, X_per_k_mod)
axis([k(1) k_per(end) 0 abs_max])
set(gca, "color", bck_clr)
ylabel('| X_{per} [k] |')
title({'Periodic signal', 'X_{per} [k]'})
```

```
139 %
    subplot(2,4,8)
141 stem(k_per, X_per_k_deg)
    axis([k_per(1) k_per(end) -200 200 ])
143 set(gca, "color", bck_clr)
    ylabel('\angle X_{per} [k]    (degree)')
145 xlabel('k')

147 %%%%%%%%%% %%%%%%%%%%

149 %
    % EOF
151 %
```

6.3 DTFT

- O Código 6.2 implementa o cálculo da DTFT de um sinal discreto não periódico, com simetria par.

Código 6.2: DTFT de sequência com simetria par.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: dtft_even_seq.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                       %
% Titulo: Demo de DTFT                  %
%         de seqüência com simetria par %
%                                       %
% Autor : Alexandre Santos de la Vega   %
% Data  : / 2010_02 /                   %
%                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


% Limpeza do ambiente
clear all
close all

%
n   = -10:10;
x1n = (-2<=n) & (n<=2);
x2n = (0<=n) & (n<=4);


%
k = -1:.01:1;
W = pi*k;

%
X1W = ( 1 + cos(W) + cos(2*W) );
X1Wmod = abs(X1W);
X1Wang = angle(X1W);

X2W = ( 1 + cos(W) + cos(2*W) ) .* exp(-j*4*W);
X2Wmod = abs(X2W);
X2Wang = unwrap(angle(X2W));


%
figure(1)
%
subplot(3,2,1)
stem(n,x1n)
ylabel('x(n)')
xlabel('n')
```

```
    title('DTFT de sinal com simetria par')
53  %
    subplot(3,2,3)
55  plot(k,X1Wmod)
    ylabel('|X(\Omega)|')
57  xlabel('\Omega / \pi')
    %
59  subplot(3,2,5)
    plot(k,X1Wang)
61  ylabel('\angle X(\Omega)')
    xlabel('\Omega / \pi')
63
    %
65  subplot(3,2,2)
    stem(n,x2n)
67  ylabel('x(n)')
    xlabel('n')
69  title('DTFT de sinal com simetria par')
    %
71  subplot(3,2,4)
    plot(k,X2Wmod)
73  ylabel('|X(\Omega)|')
    xlabel('\Omega / \pi')
75  %
    subplot(3,2,6)
77  plot(k,X2Wang)
    ylabel('\angle X(\Omega)')
79  xlabel('\Omega / \pi')

81
    %
83  % EOF
    %
```

6.4 DTFS × DTFT × DFT

- O Código 6.3 ilustra o relacionamento entre DTFS, DTFT e DFT através de cálculos envolvendo uma sequência gate retangular.

Código 6.3: Cálculo de DTFS, DTFT e DFT, para sequência gate retangular.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: dtfs_dtft_dft_gate_retangular.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
% Titulo: Calculo de DTFS, DTFT e DFT                                           %
%         de uma sequencia gate retangular.                                     %
%                                                                               %
% Autor : Alexandre Santos de la Vega                                           %
% Data  : / 2011_11_29 /                                                        %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


% limpeza do ambiente de trabalho
%  variaveis
clear all
%  janelas
close all


% definicao dos parametros
Ng = 3;      % comprimento do gate = 2Ng + 1
%
Np = 20;     % periodo fundamental de repeticao
%
Nv = 2*Np;
n = -Nv:Nv; % janela de visualizacao
%


% definicao da sequencia nao periodica
xn = gate_retang_unitario(Ng,n);


% definicao da sequencia periodica
xp = gate_retang_unitario(Ng,n);
l=1;
while ((l*Np - Ng) < Nv),
    xp = xp + gate_retang_unitario(Ng,(n + l*Np));
    xp = xp + gate_retang_unitario(Ng,(n - l*Np));
    l = l + 1;
end
```

```
52 % definicao do Omega_k = dW
   dW = (2*pi/Np);
54

56 % definicao da DTFS
   k  = n;
58 ind_0 = find(k==0);
   k(ind_0)= Np; % evita divisao por zero,
60               % pois sin(2*pi) ~= sin(0)
                 % por erro de calculo numerico
62 Wk = k*dW;
   xkp = ( sin((2*Ng + 1)*(Wk/2)) ./ sin(Wk/2) ) / Np;
64 k(ind_0)= 0;  % restaura valor original para vetor(0)

66
   % definicao da DTFT
68 l  = -Nv:(Np/500):Nv;
   ind_0 = find(l==0);
70 l(ind_0)= Np; % evita divisao por zero,
                 % pois sin(2*pi) ~= sin(0)
72               % por erro de calculo numerico
   W  = l*dW;
74 xW = ( sin((2*Ng + 1)*(W/2)) ./ sin(W/2) );
   l(ind_0)= 0;  % restaura valor original para vetor(0)
76

78 % definicao da DFT
   xk = zeros(1,length(k));
80 krange = (1:Np) + Nv;
   xk(krange) = Np * xkp(krange);
82

84 %%%%%%%%%%%%%%%%%%%%%%%%

86
   % ilustra resultados
88
   %axis([XMIN XMAX YMIN YMAX])
90
   figure(1)
92 %
   %subplot(2,3,1)
94 subplot(3,2,1)
   stem(n,xp,'k')
96 ylabel('x_p[n]')
   xlabel('n')
98 title('Sinal periódico')
   %
100 %subplot(2,3,2)
   subplot(3,2,3)
102 stem(n,xn,'k')
   ylabel('x[n]')
104 xlabel('n')
   title('Sinal não periódico')
106 %
   %subplot(2,3,3)
108 subplot(3,2,5)
   stem(n,xn,'k')
110 ylabel('x [n]')
```

```
    xlabel('n')
112 title('Sinal não periódico')
    %
114 %subplot(2,3,4)
    subplot(3,2,2)
116 stem(k,xkp,'k')
    ylabel('X_p[k]')
118 xlabel('k')
    title('DTFS de x_p[n]')
120 v = axis;
    v(1) = min(k);
122 v(2) = max(k);
    v(3) = min(xkp);
124 v(4) = max(xkp);
    axis(v)
126 %
    %subplot(2,3,5)
128 subplot(3,2,4)
    plot(W,xW,'k')
130 ylabel('X(e^{j \Omega})')
    xlabel('\Omega')
132 title('DTFT de x[n]')
    v = axis;
134 v(1) = min(W);
    v(2) = max(W);
136 v(3) = min(xW);
    v(4) = max(xW);
138 axis(v)
    %
140 %subplot(2,3,6)
    subplot(3,2,6)
142 stem(k,xk,'k')
    ylabel('X [k]')
144 xlabel('k')
    title('N_p-point DFT de x [n]')
146 v = axis;
    v(1) = min(k);
148 v(2) = max(k);
    v(3) = min(xk);
150 v(4) = max(xk);
    axis(v)
152

154 %%%%%%%%%%%%%%%%%%%%%%%%%

156

    %%
158 %% Para conferir o resultado da equacao
    %% com o resultado da funcao fft(.)
160 %%
    %fg = fft(xp(1:20));
162 %%
    %figure(2)
164 %stem(0:19, (abs(fg) .* real(exp(j*angle(fg)))))
    %ylabel('X [k]')
166 %xlabel('k')
    %title('N_p-point DFT de x [n], calculada pela função fft(x)')
168 %axis(v)
    %%
```

```
170
172 %%%%%%%%%%%%%%%%%%%%%%%%
174
    %
176 % EOF
    %
```

6.5 DTFT × DFT

- O Código 6.4 ilustra o cálculo da DTFT de um sinal discreto não periódico, através da interpolação dos coeficientes da DFT, calculados para a extensão periódica de tal sinal.

Código 6.4: DTFT calculada por interpolação da DFT.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: dtft_interp_dft.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DTFT interpolated from DFT                                                    %
% Author: Alexandre S. de la Vega                                               %
% Version: /2010_12_14/                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


% clear workspace
clear all
close all


% define input sequence
x_bas = rand(1,10);
N = length(x_bas);


% calculate DTFS coeficients
a = zeros(1,N);
for k = 0:(N-1)
    ind = k+1;
    for n = 0:(N-1)
        a(ind) = a(ind) + ( x_bas(n+1) * exp(-j*k*((2*pi)/N)*n) );
    end
end
a_mod = abs(a);
a_deg = (angle(a)*180)/pi;


% calculate DFT coeficients
X_k = fft(x_bas,N);
X_k_mod = abs(X_k);
X_k_deg = (angle(X_k)*180)/pi;


% calculate DTFT interpolation
m = 0:199;
M = length(m);
Omega_step = (2*pi)/M;
Omega = Omega_step .* m;

X_omega = zeros(1,M);
```

```
for k = 0:(N-1)
    sin_ratio_param = ( (N*Omega) - (k*2*pi) ) / 2;
    sin_ratio = sin(sin_ratio_param) ./ sin(sin_ratio_param/N);
    X_partial = a(k+1) .* sin_ratio .*    ...
                exp(-j.*(Omega-(k*((2*pi)/N))).*((N-1)/2));
    X_omega = X_omega + X_partial;
end
X_omega = X_omega / N;

X_omega_mod = abs(X_omega);
X_omega_deg = (angle(X_omega)*180)/pi;


% draw sequences
figure(1)

n = 0:(N-1);
k = 0:(N-1);

subplot(4,2,1)
stem(n, x_bas)
ylabel('x[n]')
xlabel('n')
title('Representation of a nonperiodic signal (x[n]):
       Time (n)  X  Frequency (k)')

%x_per = [x_bas x_bas x_bas];
%n_per = 0:(length(x_per)-1);
%subplot(4,2,2)
%stem(n_per, x_per)
%ylabel('x_{per}[n]')
%xlabel('n')

subplot(4,2,3)
stem(k, a_mod,'r')
ylabel('| a_{k} |')
xlabel('k')
title('DFT representation of a periodic extension (\Delta k = 2*\pi / N)')

subplot(4,2,4)
stem(k, a_deg,'r')
ylabel('\angle a_{k} (degree)')
xlabel('k')

subplot(4,2,5)
stem(k, X_k_mod,'r')
ylabel('| X_{k} |')
xlabel('k')
title('FFT representation of a periodic extension (\Delta k = 2*\pi / N)')

subplot(4,2,6)
stem(k, X_k_deg,'r')
ylabel('\angle X_{k} (degree)')
xlabel('k')

subplot(4,2,7)
plot(Omega, X_omega_mod)
hold on
stem(k*(2*pi/N),X_k_mod,'r')
```

```
111 ylabel('| X(e^{ j \Omega}) |')
    xlabel('Omega (rad)')
113 title('Interpolated DTFT from the DFT representation (\Delta k = 2*\pi / N)')
    AV = axis;
115 axis([AV(1) (2*pi) AV(3) AV(4)])

117 subplot(4,2,8)
    plot(Omega, X_omega_deg)
119 hold on
    stem(k*(2*pi/N),X_k_deg,'r')
121 ylabel('\angle X(e^{ j \Omega}) (degree)')
    xlabel('Omega (rad)')
123 AV = axis;
    axis([AV(1) (2*pi) AV(3) AV(4)])
125

127 %
    % EOF
129 %
```

6.6 DFT × *Leakage* ou *Smearing*

- O Código 6.5 ilustra o fenômeno de *leakage* ou *smearing* no cálculo da DFT.

Código 6.5: Fenômeno de *leakage* no cálculo da DFT.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: dft_leakage_demo.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                          %
% Titulo: Demo de DFT de sequencia senoidal %
%                                          %
% Autor : Alexandre Santos de la Vega      %
% Data  : / 01/07/2k9 / 2010_12_14 /       %
%                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


% Limpeza do ambiente
clear all
close all


% Parametros da sequencia senoidal original
A0 = 2;
F0 = 100;
T0 = 1/F0;
P0 = 0;


% Parametros da amostragem
SamplingFactor = 128;        % Numero de pontos por periodo
Fs = (SamplingFactor * F0); % Sampling frequency
Ts = 1/Fs;

Nper = 4;
Nppp = (T0/Ts);         % Numero de pontos por periodo
Ntot = (Nper * Nppp); % Total de pontos amostrados


% Montagem da sequencia amostrada
n = 0:(Ntot-1);
x = A0*cos((2*pi*F0*(n*Ts)) + P0);


% Calculo da DFT sem leakage
NPfft = Ntot; % Numero de pontos da DFT sem leakage
k = 0:(NPfft-1);

X = fft(x,NPfft);

Xmod = abs(X);
```

```
Xang_rad = angle(X);
Xang_deg = Xang_rad*180/pi;


% Calculo da DFT com leakage, devido a nao casamento de periodos
NPfftlp = Ntot*((2*Nper)-1)/(2*Nper); % Numero de pontos da DFT com leakage
klp = 0:(NPfftlp-1);

Xlp = fft(x,NPfftlp);

Xlpmod = abs(Xlp);
Xlpang_rad = angle(Xlp);
Xlpang_deg = Xlpang_rad*180/pi;


% Calculo da DFT com leakage, devido a insercao de zeros
NPfftl = 3*Ntot; % Numero de pontos da DFT com leakage

nl = 0:(NPfftl-1);
ZeroPaddingVector = zeros(1,(NPfftl-NPfft));
xl = [x, ZeroPaddingVector];

kl = 0:(NPfftl-1);
Xl = fft(x,NPfftl);

Xlmod = abs(Xl);
Xlang_rad = angle(Xl);
Xlang_deg = Xlang_rad*180/pi;


%
% Graficos
%

% Grafico da DFT com leakage, devido a insercao de zeros
figure(1)

subplot(2,2,1)
stem(n,x)
title('Sinal original: x[n] = A_0 cos(2\piF_0nT_s + \phi_0)')
xlabel('n')
ylabel('x[n]')
AV = axis;
axis([AV(1) NPfft AV(3) AV(4)])

subplot(2,2,2)
stem(k,Xmod)
title('Módulo da DFT de x[n]')
xlabel('k')
ylabel('| X[k] |')
AV = axis;
axis([AV(1) NPfft AV(3) AV(4)])

subplot(2,2,4)
stem(k,Xang_deg)
title('Ângulo de fase da DFT de x[n]')
xlabel('k')
ylabel('\angle X[k]')
AV = axis;
```

```
113 axis([AV(1) NPfft AV(3) AV(4)])

115
    % Grafico da DFT com leakage, devido a nao casamento de periodos
117 figure(2)

119 subplot(2,2,1)
    stem(klp,x(klp+1))
121 title('Sinal original: x[n] = A_0 cos(2\piF_0nT_s + \phi_0)')
    xlabel('n')
123 ylabel('x[n]')
    AV = axis;
125 axis([AV(1) NPfftlp AV(3) AV(4)])

127 subplot(2,2,2)
    stem(klp,Xlpmod)
129 title('Módulo da DFT de x[n]')
    xlabel('k')
131 ylabel('| X[k] |')
    AV = axis;
133 axis([AV(1) NPfftlp AV(3) AV(4)])

135 subplot(2,2,4)
    stem(klp,Xlpang_deg)
137 title('Ângulo de fase da DFT de x[n]')
    xlabel('k')
139 ylabel('\angle X[k]')
    AV = axis;
141 axis([AV(1) NPfftlp AV(3) AV(4)])

143
    % Grafico da DFT com leakage, devido a insercao de zeros
145 figure(3)

147 subplot(2,2,1)
    stem(nl,xl)
149 title('Sinal original: x[n] = A_0 cos(2\piF_0nT_s + \phi_0)')
    xlabel('n')
151 ylabel('x[n]')
    AV = axis;
153 axis([AV(1) NPfftl AV(3) AV(4)])

155 subplot(2,2,2)
    stem(kl,Xlmod)
157 title('Módulo da DFT de x[n]')
    xlabel('k')
159 ylabel('| X[k] |')
    AV = axis;
161 axis([AV(1) NPfftl AV(3) AV(4)])

163 subplot(2,2,4)
    stem(kl,Xlang_deg)
165 title('Ângulo de fase da DFT de x[n]')
    xlabel('k')
167 ylabel('\angle X[k]')
    AV = axis;
169 axis([AV(1) NPfftl AV(3) AV(4)])

171 %
```

```
    % EOF
173 %
```

6.7 Aceleração do cálculo da DFT

6.7.1 Cálculo da DFT de sequências reais empregando sequências complexas

- O Código 6.6 define uma função para calcular as N-DFTs de duas sequências reais através do cálculo da N-DFT de uma sequência complexa.
- O Código 6.7 ilustra o cálculo das N-DFTs de duas sequências reais através do cálculo da N-DFT de uma sequência complexa.
- O Código 6.8 ilustra o cálculo da 2N-DFT de uma sequência real através do cálculo da N-DFT de uma sequência complexa.

Código 6.6: Cálculo das N-DFTs de duas sequências reais através do cálculo da N-DFT de uma sequência complexa.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: cs_fft.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [Gk,Hk] = cs_fft(g,h)

%
% CS_FFT Complex sequence FFT
%    CS_FFT(g,h) returns G[k] and H[k] by using
%                a single FFT operation.
%

x = g + (j*h);

X_fft = fft(x);

X_aux = [X_fft , X_fft(1)];
X_fft_conj_refl_circ = conj( fliplr(X_aux(2:end)) );

Gk = (X_fft + X_fft_conj_refl_circ) / (2);
Hk = (X_fft - X_fft_conj_refl_circ) / (2*j);

end

%
% EOF
%
```

Código 6.7: Exemplo de cálculo das N-DFTs de duas sequências reais através do cálculo da N-DFT de uma sequência complexa.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: two_seqs_complex_fft.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                            %
% Titulo: Calculo de FFT                     %
%         de uma sequencia complexa,         %
%         formada por duas sequencias reais  %
%         distintas: x[n] = g[n] + j h[n]    %
%                                            %
% Autor : Alexandre Santos de la Vega        %
% Data  : / 2011_11_24 /                     %
%                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


% limpeza do ambiente de trabalho
%   variaveis
clear all
%   janelas
close all


% definicao dos parametros
N  = 40;
n = 0:(N-1);

% definicao das sequencias de entrada
g = zeros(1,N);
g(1:7) = 1;

h = zeros(1,N);
h(1:11) = 1;


% calculo das FFTs isoladamente
tic
%
G_fft = fft(g);
H_fft = fft(h);
%
t_2r_fft = toc;


% calculo da FFT com sequencia complexa
tic
%
[G_cs_fft,H_cs_fft] = cs_fft(g,h);
%
t_1c_fft = toc;


%%%%%%%%%%%%%%%%%%%%%%


% ilustra resultados

%AXIS([XMIN XMAX YMIN YMAX])
```

```
65
figure(1)
67 %
subplot(4,2,1)
69 stem(n,abs(G_fft),'k')
ylabel('|G[k]|')
71 title('Módulos calculados por duas N-FFTs de sequências reais')
%
73 subplot(4,2,2)
stem(n, angle(G_fft),'k')
75 ylabel('\angle G[k]')
title('Ângulos de fase calculados por duas N-FFTs de sequências reais')
77 %
subplot(4,2,3)
79 stem(n,abs(H_fft),'k')
ylabel('|H[k]|')
81 %
subplot(4,2,4)
83 stem(n, angle(H_fft),'k')
ylabel('\angle H[k]')
85 %
subplot(4,2,5)
87 stem(n,abs(G_cs_fft),'r')
ylabel('|G[k]|')
89 title('Módulos calculados por uma N-FFT de sequência complexa')
%
91 subplot(4,2,6)
stem(n, angle(G_cs_fft),'r')
93 ylabel('\angle G[k]')
title('Ângulos de fase calculados por uma N-FFT de sequência complexa')
95 %
subplot(4,2,7)
97 stem(n,abs(H_cs_fft),'r')
ylabel('|H[k]|')
99 xlabel('k')
%
101 subplot(4,2,8)
stem(n, angle(H_fft),'r')
103 ylabel('\angle H[k]')
xlabel('k')
105


107 figure(2)
%
109 subplot(2,2,1)
stem(n,abs(G_fft),'k')
111 hold on
stem(n,abs(G_cs_fft),'r')
113 ylabel('|G[k]|')
title('Comparação dos módulos')
115 %
subplot(2,2,2)
117 stem(n,angle(G_fft),'k')
hold on
119 stem(n,angle(G_cs_fft),'r')
ylabel('\angle G[k]')
121 title('Comparação dos ângulos de fase')
%
123 subplot(2,2,3)
```

```
stem(n,abs(H_fft),'k')
hold on
stem(n,abs(H_cs_fft),'r')
ylabel('|H[k]|')
xlabel('k')
%
subplot(2,2,4)
stem(n,angle(H_fft),'k')
hold on
stem(n,angle(H_cs_fft),'r')
ylabel('\angle H[k]')
xlabel('k')


%%%%%%%%%%%%%%%%%%%


%
% EOF
%
```

Código 6.8: Exemplo de cálculo da 2N-DFT de uma sequência real através do cálculo da N-DFT de uma sequência complexa.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: one_seq_complex_fft.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                   %
% Titulo: Calculo de FFT            %
%         de uma sequencia complexa, %
%         formada por duas sequencias reais %
%         provenientes de uma sequencia: %
%         x[n] = g[n] + j h[n]      %
%         g[n] = v[2n]              %
%         h[n] = v[2n+1]            %
%                                   %
% Autor : Alexandre Santos de la Vega %
% Data  : / 2011_11_24 /            %
%                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


% limpeza do ambiente de trabalho
%  variaveis
clear all
%  janelas
close all


% definicao dos parametros
```

```
M = 40;
m = 0:(M-1);
ind_m = (m + 1);
m_par = (2*m);
m_impar = (2*m) + 1;
ind_m_par = m_par + 1;
ind_m_impar = m_impar + 1;
%
N = 2*M;
n = 0:(N-1);


% definicao da sequencia de entrada
v = zeros(1,N);
v(1:11) = 1;

% calculo da FFT isoladamente
tic
%
V_fft = fft(v);
%
t_2r_fft = toc;


% calculo da FFT com sequencia complexa
tic
%
g = v(ind_m_par);
h = v(ind_m_impar);
%
[G_cs_fft,H_cs_fft] = cs_fft(g,h);
%
V_cs_fft(ind_m) = G_cs_fft + (exp(-j*(pi/M)*m) .* H_cs_fft);
V_cs_fft(ind_m + M) = G_cs_fft + (exp(-j*(pi/M)*(m+M)) .* H_cs_fft);
%
t_1c_fft = toc;


%%%%%%%%%%%%%%%%%%%%%%%%%

% ilustra resultados

%AXIS([XMIN XMAX YMIN YMAX])

figure(1)
%
subplot(3,2,1)
stem(n,abs(V_fft),'k')
ylabel('|V[k]|')
title('Módulo calculado por uma 2N-FFT de sequência real')
%
subplot(3,2,2)
stem(n, angle(V_fft),'k')
ylabel('\angle V[k]')
title('Ângulo de fase calculado por uma 2N-FFT de sequência real')
%
subplot(3,2,3)
stem(n,abs(V_cs_fft),'r')
```

```
93  ylabel('|V[k]|')
    title('Módulo calculado por uma N-FFT de sequência complexa')
95  %
    subplot(3,2,4)
97  stem(n, angle(V_cs_fft),'r')
    ylabel('\angle V[k]')
99  title('Ângulo de fase calculado por uma N-FFT de sequência complexa')
    %
101 subplot(3,2,5)
    stem(n,abs(V_fft),'k')
103 hold on
    stem(n,abs(V_cs_fft),'r')
105 ylabel('|V[k]|')
    title('Comparação dos módulos')
107 xlabel('k')
    %
109 subplot(3,2,6)
    stem(n,angle(V_fft),'k')
111 hold on
    stem(n,angle(V_cs_fft),'r')
113 ylabel('\angle V[k]')
    title('Comparação dos ângulos de fase')
115 xlabel('k')

117
    %%%%%%%%%%%%%%%%%%%%%%%%%
119

121 %
    % EOF
123 %
```

6.7.2 FFT

- O Código 6.9 ilustra a comparação entre o número de operações da DFT e de um tipo de FFT.

Código 6.9: Comparação entre o número de operações da DFT e de um tipo de FFT.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Arquivo: dft_fft_ops_comparison.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                           %
% Titulo: Comparacao do numero de operacoes %
%         entre                             %
%         DFT por equacao original          %
%         e                                 %
%         FFT implementada no simulador     %
%                                           %
% Autor : Alexandre Santos de la Vega       %
% Data  : / 2011_11_17 /                    %
%                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%


% limpeza do ambiente de trabalho
%  variaveis
clear all
%  janelas
close all


% definicao dos parametros
N  = 2:1500;
m  = 1:10;
Np = 2.^(m);


% calculo do numero de operacoes para DFT
mc_dft = N.^2;
ac_dft = N.*(N-1);
%
mcp_dft = mc_dft(Np);
acp_dft = ac_dft(Np);


% calculo do numero de operacoes para DFT
mc_fft = (N./2).*log2(N);
ac_fft = N.*log2(N);
%
mcp_fft = mc_fft(Np);
acp_fft = ac_fft(Np);

```

```
% calculo da aceleracao
mc = mc_dft ./ mc_fft;
ac = ac_dft ./ ac_fft;
%
mcp = mc(Np);
acp = ac(Np);


%%%%%%%%%%%%%%%%%%%%%%%%%


% ilustra resultados
figure(1)
%
%AXIS([XMIN XMAX YMIN YMAX])
max_mc_ac_fft = ceil(max(max(mc_fft),max(ac_fft)));
%
subplot(4,2,1)
stem(Np,mcp_dft,'k')
hold on
plot(N,mc_dft,'k')
ylabel('N^2')
title('Number of complex multiplications: DFT')
%
subplot(4,2,2)
stem(Np,acp_dft,'k')
hold on
plot(N,ac_dft,'k')
ylabel('N(N-1)')
title('Number of complex additions: DFT')
%
subplot(4,2,3)
stem(Np,mcp_fft,'r')
hold on
plot(N,mc_fft,'r')
v = AXIS;
v(4) = max_mc_ac_fft;
AXIS(v)
ylabel('N/2 log_2(N)')
title('Number of complex multiplications: FFT')
%
subplot(4,2,4)
stem(Np,acp_fft,'r')
hold on
plot(N,ac_fft,'r')
v = AXIS;
v(4) = max_mc_ac_fft;
AXIS(v)
ylabel('N log_2(N)')
title('Number of complex additions: FFT')
%
subplot(4,2,5)
stem(Np,mcp_dft,'k')
hold on
plot(N,mc_dft,'k')
%
stem(Np,mcp_fft,'r')
plot(N,mc_fft,'r')
title('Number of complex multiplications: DFT and FFT')
```

```
    %
113 subplot(4,2,6)
    stem(Np,acp_dft,'k')
115 hold on
    plot(N,ac_dft,'k')
117 %
    stem(Np,acp_fft,'r')
119 plot(N,ac_fft,'r')
    title('Number of complex additions: DFT and FFT')
121 %
    subplot(4,2,7)
123 stem(Np,mcp,'b')
    hold on
125 plot(N,mc,'b')
    title('Number-of-complex-multiplications gain')
127 xlabel('N')
    %
129 subplot(4,2,8)
    stem(Np,acp,'b')
131 hold on
    plot(N,ac,'b')
133 title('Number-of-complex-additions gain')
    xlabel('N')
135

137 %%%%%%%%%%%%%%%%%%%%%%%%%

139
    %
141 % EOF
    %
```

Referências bibliográficas

[Ant86] A. Antoniou. *Digital Filters: Analysis and Design*. Tata McGraw-Hill, New Delhi, India, 2nd reprint edition, 1986.

[Cad73] J. A. Cadzow. *Discrete-Time Systems: An Introduction with Interdisciplinary Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[DdSN10] P. S. R. Diniz, E. A. B. da Silva, and S. Lima Netto. *Digital Signal Processing: System Analysis and Design*. Cambridge University Press, Cambridge, UK, 2nd edition, 2010.

[Jac96] L. B. Jackson. *Digital Filters and Signal Processing - with MATLAB exercises*. Kluwer Academic Publishers, 3rd edition, 1996.

[KD04] H. Kopka and P. W. Daly. *A Guide to LaTeX and Electronic Publishing*. Addison-Wesley, Harlow, England, 4th edition, 2004.

[MG04] F. Mittelbach and M. Goosens. *The LaTeX Companion*. Addison-Wesley, Boston, MA, USA, 2th edition, 2004.

[Mit98] S. K. Mitra. *Digital Signal Processing: A Computer-Based Approach*. McGraw-Hill, New York, NY, 1998.

[OS75] A. V. Oppenheim and R. W. Schafer. *Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1975.

[OWY83] A. V. Oppenheim, A. S. Willsky, and I. T. Young. *Signals and Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1983.

[PL76] A. Peled and B. Liu. *Digital Signal Processing: Theory, Design and Implementation*. John Wiley, New York, NY, 1976.

[PM06] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing: Principles, Algorithms and Applications*. Prentice Hall, 4th edition, 2006.

[Rob09] M. J. Roberts. *Fundamentos em Sinais e Sistemas*. McGraw-Hill, São Paulo, SP, 2009.

[SDD84] W. D. Stanley, G. R. Dougherty, and R. Dougherty. *Signals and Systems*. Prentice-Hall, Reston, Virginia, 2nd edition, 1984.

[She95] K. Shenoi. *Digital Signal Processing in Telecommunications*. Prentice-Hall PTR, Upper Saddle River, NJ, 1995.

[SK89] R. D. Strum and D. E. Kirk. *First Principles of Discrete Systems and Digital Signal Processing*. Addison-Wesley, Massachusetts, 1989.