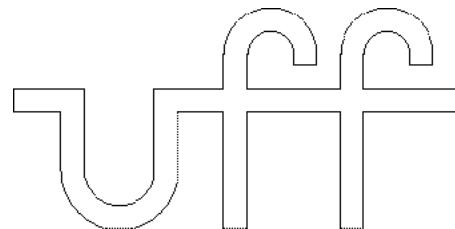


**Apostila
de
Códigos de Programas Demonstrativos
para
Fundamentos de
Processamento Digital de Sinais**

(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	de la Vega, Alexandre Santos
(*)	
D278	Apostila com Códigos de Programas Demonstrativos para Processamento Digital de Sinais / Alexandre Santos de la Vega. – Niterói:
(*)	
2025	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 intitulado Apostila de Teoria para Fundamentos de Processamento Digital de Sinais.
2. O conteúdo prático pode ser encontrado no volume intitulado 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 intitulado 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 intitulado 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 intitulado 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 intitulado 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 L^AT_EX [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

Prefácio	v
Agradecimentos	vii
Apresentação do material didático	ix
Teoria abordada no material didático	xiii
Objetivos da disciplina	xvii
Sumário	xix

I Introdução 1

1 Ambiente de simulação matemática 3
1.1 Introdução 3
1.2 Aspectos gerais sobre o ambiente 3
1.2.1 Definições 3
1.2.2 Interação com usuário 4
1.2.3 Interpretador de comandos 4
1.2.4 <i>Namespace</i> e <i>workspace</i> 5
1.2.5 Tipos de dados 5
1.2.6 Estruturas de dados 6
1.2.7 Funções básicas dedicadas a matrizes 6
1.2.8 Códigos 7
1.3 Representação gráfica bidimensional (curvas) 25
1.3.1 Representação gráfica discreta de função unidimensional 25
1.3.2 Representação gráfica contínua de função unidimensional 25
1.3.3 Funções comumente utilizadas 25
1.3.4 Códigos 25
1.4 Representação gráfica tridimensional (curvas e superfícies) 33
1.4.1 Representação gráfica discreta de função bidimensional 33
1.4.2 Representação gráfica contínua de função bidimensional 33
1.4.3 Funções comumente utilizadas 33
1.4.4 Códigos 33
1.5 Representação gráfica tridimensional (imagens) 40
1.5.1 Representação gráfica discreta de função bidimensional 40
1.5.2 Representações de imagens no aplicativo 40

1.5.3	Estruturas de dados para imagens no aplicativo	41
1.5.4	Amostragem e exibição das imagens	42
1.5.5	Funções comumente utilizadas	43
1.5.6	Códigos	43
1.6	Simulação de uma equação de diferença	52
1.6.1	Equação de diferença	52
1.6.2	Simulação de uma equação de diferença no aplicativo	52
1.6.3	Códigos	53
1.7	Exercícios propostos	56
1.7.1	Vetores: construção e manipulação	56
1.7.2	Matrizes: construção e manipulação	57
1.7.3	Matrizes: cálculos	58
1.7.4	Modelagem matricial	59
2	Conceitos básicos	61
2.1	Introdução	61
2.2	Tipos de sinais	61
2.3	Amostragem	65
2.4	Arquitetura de sistemas de processamento digital	80
II	Sinais e sistemas no domínio do tempo	85
3	Sinais no domínio do tempo	87
3.1	Introdução	87
3.2	Tipos de sequências	87
3.2.1	Sistema numérico	87
3.2.2	Comprimento	89
3.2.3	Simetria	91
3.3	Operações básicas sobre sequências	94
3.4	Sequências mais comumente empregadas	119
4	Sequências exponenciais	131
4.1	Introdução	131
4.2	Características relevantes de exponenciais	131
4.3	Efeitos das características relevantes de exponenciais	150
III	Representações de um SLIT	177
5	Representações de um SLIT	179
5.1	Introdução	179
5.2	Operador de transferência	179
5.2.1	Diagrama de Pólos e Zeros (DPZ)	179
IV	Sinais e sistemas no domínio da frequência	185
6	Sinais no domínio da frequência	187
6.1	Introdução	187

6.2	DTFS	187
6.3	DTFT	191
6.4	DTFS \times DTFT \times DFT	193
6.5	DTFT \times DFT	197
6.6	DFT \times <i>Leakage</i> ou <i>Smearing</i>	200
6.7	Aceleração do cálculo da DFT	204
6.7.1	Cálculo da DFT de sequências reais empregando sequências complexas .	204
6.7.2	FFT	210

Referências bibliográficas**213**

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 forma 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 ($D_n = 1$) 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.

```

1  %%%%%%
2  %
3  % Arquivo: Aula_1_Lab_Octave_Criacao.m
4  %
5  %%%%%%
6
7  %%%%%%
8  %
9  % Titulo:
10 %
11 % Exemplos de criacao de estruturas de dados %
12 %
13 %
14 % Autor : Alexandre Santos de la Vega %
15 % Datas : /2022-01/
16 %
17 %%%%%%
18
19 % limpeza do ambiente de trabalho
20 %
21 clear all
22 close all
23
24
25 disp( ' ' )
26 disp( ' ' )
27 disp( ' ' ) =====)
28 disp( ' ' ) ===== Exemplos de criação de estruturas de dados')
29 disp( ' ' ) =====)
30 disp( ' ' )
31 disp( ' ' )
32
33 %
34 % Exemplos de escalar (matriz 1x1)
35 %
36
37 disp( ' ' )
38 disp( ' ' )
39 disp( ' ' ) =====)
40 disp( ' ' ) ===== Exemplos de escalar (matriz 1x1)')
41 disp( ' ' ) =====)
42 disp( ' ' )
43 disp( ' ' )
44
45 v = 'a'
46
47 disp( ' ' )
48 disp( 'Pressione qualquer tecla para continuar... ')
49 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 %
75 disp( ' ')
77 disp( ' ', Exemplos de vetor linha ( matriz 1xC ), )
79 disp( ' ')
81 v = 'abc'
83 disp( ' ')
disp('Pressione qualquer tecla para continuar... ')
85 pause()
87 v = [4 12 100]
89 disp( ' ')
91 pause()
93 v = [-8.5, 4, 11.3]
95 disp( ' ')
97 pause()
99 v = [3 + i*7, -2 + i*9, i*11]
101 disp( ' ')
103 pause()
105 v = 5:11
107 disp( ' ')
disp('Pressione qualquer tecla para continuar... ')

```

```
109 | pause()
111 | v = 12:3:22
113 | disp( , )
114 | disp( 'Pressione qualquer tecla para continuar...' )
115 | pause()
117 | v = 78:-5:49
119 | disp( , )
120 | disp( 'Pressione qualquer tecla para continuar...' )
121 | pause()
123 |
124 | %
125 | % Exemplos de vetor coluna (matriz Lx1)
126 | %
127 | disp( , )
128 | disp( , )
129 | disp( , =====, )
130 | disp( , Exemplos de vetor coluna (matriz Lx1) )
131 | disp( , =====, )
132 | disp( , =====, )
133 | disp( , )
135 | v = [ 'a' ; 'b' ; 'c' ]
137 | disp( , )
138 | disp( 'Pressione qualquer tecla para continuar...' )
139 | pause()
141 | v = [ 4 ; 12 ; 100 ]
143 | disp( , )
144 | disp( 'Pressione qualquer tecla para continuar...' )
145 | pause()
147 | v = [ -8.5 ; 4 ; 11.3 ]
149 | disp( , )
150 | disp( 'Pressione qualquer tecla para continuar...' )
151 | pause()
153 | v = [ 3 + i*7 ; -2 + i*9 ; i*11 ]
155 | disp( , )
156 | disp( 'Pressione qualquer tecla para continuar...' )
157 | pause()
159 | v = [ 5:11 ] ,
161 | disp( , )
162 | disp( 'Pressione qualquer tecla para continuar...' )
163 | pause()
165 | v = [ 12:3:22 ] ,
167 | disp( , )
```

```

169 | disp( 'Pressione qualquer tecla para continuar... ')
169 | pause()
171 v = [76:-5:49] '
173 disp(' ')
173 disp( 'Pressione qualquer tecla para continuar... ')
175 pause()

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

189 v = [ 'abcd'; 'efgh' ]
191 disp(' ')
191 disp( 'Pressione qualquer tecla para continuar... ')
193 pause()

195 v = [ 'ab'; 'cd' ; 'ef' ; 'gh' ]
197 disp(' ')
197 disp( 'Pressione qualquer tecla para continuar... ')
199 pause()

201 v = [4 12 100 1 ; 7 11 15 3]
203 disp(' ')
203 disp( 'Pressione qualquer tecla para continuar... ')
205 pause()

207 v = [4 12 ; 100 1 ; 7 11 ; 15 3]
209 disp(' ')
209 disp( 'Pressione qualquer tecla para continuar... ')
211 pause()

213 v = [1:6 ; 11:16 ; 21:26]
215 disp(' ')
215 disp( 'Pressione qualquer tecla para continuar... ')
217 pause()

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

```

```

227 | disp( , ) Exemplos de matriz quadrada M (matriz MxM)
228 | =====
229 | disp( , )
230 |
231 v = [ 'abcd' ; 'efgh' ; 'ijkl' ; 'mnop' ]
232 |
233 disp( , )
234 disp( 'Pressione qualquer tecla para continuar...')
235 pause()
236 |
237 v = [4 12 100 1 ; 7 11 15 3 ; 2 6 19 33 ; 8 10 44 56]
238 |
239 disp( , )
240 disp( 'Pressione qualquer tecla para continuar...')
241 pause()
242 |
243 v = [1:4 ; 11:14 ; 21:24 ; 31:34]
244 |
245 disp( , )
246 disp( 'Pressione qualquer tecla para continuar...')
247 pause()
248 |
249 %
250 %
251 % Exemplos de cell array
252 %
253 |
254 disp( , )
255 disp( , )
256 | =====
257 disp( , ) Exemplos de cell array
258 | =====
259 disp( , )
260 |
261 %
262 % Construção com atribuição
263 %
264 |
265 disp( , )
266 disp( , )
267 | =====
268 disp( , ) Construção com atribuição
269 | =====
270 disp( , )
271 v = {
272 |     1 , [2 , 3] ;
273 |     [4 ; 5] , [6 , 7 ; 8 , 9 ; 10 , 11]
274 | }
275 |
276 disp( , )
277 disp( 'Pressione qualquer tecla para continuar...')
278 pause()
279 |
280 v{1,1}
281 |
282 disp( , )
283 disp( 'Pressione qualquer tecla para continuar...')
284 pause()
285 |

```

```

287 v{1,2}
288 disp( ' ')
289 disp('Pressione qualquer tecla para continuar... ')
290 pause()
291
292 v{2,1}
293
294 disp( ' ')
295 disp('Pressione qualquer tecla para continuar... ')
296 pause()
297
298 v{2,2}
299
300 disp( ' ')
301 disp('Pressione qualquer tecla para continuar... ')
302 pause()
303 %
304 % Construção com função
305 %
306
307 disp( ' ')
308 disp( ' ')
309 disp( ' ', _____, )
310 disp( ' ', Construção com função')
311 disp( ' ', _____, )
312 disp( ' ')
313
314
315 v = cell(3,2)
316
317 disp( ' ')
318 disp('Pressione qualquer tecla para continuar... ')
319 pause()
320
321 v{1,1} = 1.1;
322 v{1,2} = 'a';
323 v{2,1} = [2 + i*3 , 4 + i*5];
324 v{2,2} = [6 + i*7 ; 8 + i*9];
325 v{3,1} = [ 'bcd' ; 'efg' ];
326 v{3,2} = [10 , 11 ; 12 , 13 ; 14 , 15]
327
328 disp( ' ')
329 disp('Pressione qualquer tecla para continuar... ')
330 pause()
331
332 v{1,1}
333
334 disp( ' ')
335 disp('Pressione qualquer tecla para continuar... ')
336 pause()
337
338 v{1,2}
339
340 disp( ' ')
341 disp('Pressione qualquer tecla para continuar... ')
342 pause()
343
344 v{2,1}

```

```

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

369 %
369 % Exemplos de structure array
371 %

373 disp( ' ')
373 disp( ' ')
375 disp( ' ', =====)
375 disp( ' ', Exemplos de structure array')
377 disp( ' ', =====)
377 disp( ' ')
379 %

381 % Construção com atribuição
381 %
383 disp( ' ')
385 disp( ' ')
385 disp( ' ', =====)
387 disp( ' ', Construção com atribuição')
387 disp( ' ', =====)
389 disp( ' ')
389 %

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

```

```

405 | v(2).situacao = 'VS'
406 |
407 | disp(' ')
408 | disp('Pressione qualquer tecla para continuar... ')
409 | pause()
410 |
411 v(3).nome      = 'Aluno 111';
412 v(3).notas     = [7.1 8.1 9.1];
413 v(3).situacao = 'Aprovado'
414 |
415 disp(' ')
416 disp('Pressione qualquer tecla para continuar... ')
417 pause()
418 |
419 v(1)
420 |
421 disp(' ')
422 disp('Pressione qualquer tecla para continuar... ')
423 pause()
424 |
425 v(2)
426 |
427 disp(' ')
428 disp('Pressione qualquer tecla para continuar... ')
429 pause()
430 |
431 v(3)
432 |
433 disp(' ')
434 disp('Pressione qualquer tecla para continuar... ')
435 pause()
436 |
437 % Construção com função
438 %
439 |
440 disp(' ')
441 disp(' ')
442 disp(' ', _____, )
443 disp(' ', Construção com função')
444 disp(' ', _____, )
445 disp(' ')
446 |
447 v = struct(      'nome' , { 'Aluno Nota Baixa' ,      , ...
448                                'Aluno Nota Mediana' ,      , ...
449                                'Aluno Nota Suficientemente alta' , ...
450                                } , ...
451                                'media' , { 3.9 , 5.9 , 6.1 } , ...
452                                'situacao' , { 'Reprovado' , 'VS' , 'Aprovado' } ...
453                                )
454 |
455 disp(' ')
456 disp('Pressione qualquer tecla para continuar... ')
457 pause()
458 |
459 v(1)
460 |
461 disp(' ')
462 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 |
22 | clear all
23 | close all
25 |
26 | disp(' ')
27 | disp(' ')
28 | disp(' ', =====,')
29 | disp(' ', Exemplos de acesso a estruturas de dados',')
30 | disp(' ', =====,')
31 | disp(' ')
33 |
34 | %
35 | % Exemplos de vetor linha (matriz 1xC)
36 | %
37 |

```

```

39 | disp( ' ')
| disp( ' ')
| disp( , )
41 | =====, )
| Exemplos de vetor linha (matriz 1xC) ')
| =====, )
43 | disp( ' ')
45 vl = 1:2:13
47 disp( ' ')
disp( 'Pressione qualquer tecla para continuar... ')
49 pause()
51 vl(3)
53 disp( ' ')
disp( 'Pressione qualquer tecla para continuar... ')
55 pause()
57 vl([2 , 5])
59 disp( ' ')
disp( 'Pressione qualquer tecla para continuar... ')
61 pause()
63 vl(4:6)
65 disp( ' ')
disp( 'Pressione qualquer tecla para continuar... ')
67 pause()
69 %
71 % Exemplos de vetor coluna (matriz Lx1):
% 
73
75 | disp( ' ')
| disp( ' ')
| =====, )
77 | Exemplos de vetor coluna (matriz Lx1) ')
| =====, )
79 | disp( ' ')
81 vc = [2:2:16] '
83 disp( ' ')
disp( 'Pressione qualquer tecla para continuar... ')
85 pause()
87 vc(6)
89 disp( ' ')
disp( 'Pressione qualquer tecla para continuar... ')
91 pause()
93 vc([1 , 4])
95 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( ' ')
113 disp( ' ') =====
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

```

```

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

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

179 disp(' ')
179 disp(' ')
181 disp(' ')
181 =====,')
183 disp(' ') ===== Exemplos de matriz tridimensional (matriz LxCxP)')
183 disp(' ')
183 =====,')
185 disp(' ')
187 mlcP = randn(6,4,3)
187 |
187 disp(' ')
189 disp('Pressione qualquer tecla para continuar... ')
189 pause()
191 |
191 mlcP(5,3,2)
193 |
193 disp(' ')
195 disp('Pressione qualquer tecla para continuar... ')
195 pause()
197 |
197 % ( L*C*(p-1) ) + ( L*(c-1) ) + l = 
199 % ( 6*4*(2-1) ) + ( 6*(3-1) ) + 5 = 24 + 12 + 5 = 41
199 %
201 mlcP(41)
203 |
203 disp(' ')
205 disp('Pressione qualquer tecla para continuar... ')
205 pause()
207 mlcP(2, 1:3, 1)
209 |
209 disp(' ')
211 disp('Pressione qualquer tecla para continuar... ')
211 pause()
213 mlcP(2:5, 3, 2)

```

```

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

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

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

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

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

231 %
232 %
233 % Exemplos de cell array
234 %

235 disp( ' ')
236 disp( ' ')
237 disp( ' _____')
238 disp( ' Exemplos de cell array')
239 disp( ' _____')
240 disp( ' ')
241 disp( ' ')

243 %
244 %
245 % Construção com atribuição
246 %

247 disp( ' ')
248 disp( ' ')
249 disp( ' _____')
250 disp( ' Construção com atribuição')
251 disp( ' _____')
252 disp( ' ')
253 c = { 1 , [2 , 3] ;
254 [4 ; 5] , [6 , 7 ; 8 , 9 ; 10 , 11]
255 }

257 disp( ' ')
258 disp( 'Pressione qualquer tecla para continuar...')
259 pause()

261 c{1,1}
262 %

263 disp( ' ')
264 disp( 'Pressione qualquer tecla para continuar...')
265 pause()

267 c{1,2}
268 %

269 disp( ' ')
270 disp( 'Pressione qualquer tecla para continuar...')
271 pause()
272 %
273

```

```

275 | c{2,1}
275 |
277 | disp(' ')
277 | disp('Pressione qualquer tecla para continuar... ')
277 | pause()
279 |
281 | c{2,2}
281 |
283 | disp(' ')
283 | disp('Pressione qualquer tecla para continuar... ')
283 | pause()
285 |
287 | c{1,2}(1,1)
287 |
289 | disp(' ')
289 | disp('Pressione qualquer tecla para continuar... ')
289 | pause()
291 |
293 | c{2,1}(2,1)
293 |
295 | disp(' ')
295 | disp('Pressione qualquer tecla para continuar... ')
295 | pause()
297 |
299 | c{2,2}(2,:)
299 |
301 | disp(' ')
301 | disp('Pressione qualquer tecla para continuar... ')
301 | pause()
303 |

305 %
305 % Exemplos de structure array
307 %

309 | disp(' ')
311 | disp(' ')
311 | disp(''=====','')
311 | disp(''Exemplos de structure array','')
313 | disp(''=====','')
313 | disp(' ')
315 %

317 %
317 % Construção com atribuição
317 %

319 | disp(' ')
321 | disp(' ')
321 | disp(''=====','')
323 | disp(''Construção com atribuição','')
323 | disp(''=====','')
325 | disp(' ')
325 |

327 |
329 | s.nome = 'Aluno 1';
329 | s.notas = [1.1 2.1 3.1];
329 | s.situacao = 'Reprovado'
331 | disp(' ')

```

```
333 | disp( 'Pressione qualquer tecla para continuar... ')
334 | pause()
335 | s(2).nome      = 'Aluno 11';
336 | s(2).notas     = [4.1 5.1 6.1];
337 | s(2).situacao  = 'VS'
338 |
339 | disp( ' ')
340 | disp( 'Pressione qualquer tecla para continuar... ')
341 | pause()
342 |
343 | s(3).nome      = 'Aluno 111';
344 | s(3).notas     = [7.1 8.1 9.1];
345 | s(3).situacao  = 'Aprovado'
346 |
347 | disp( ' ')
348 | disp( 'Pressione qualquer tecla para continuar... ')
349 | pause()
350 |
351 | s(1)
352 |
353 | disp( ' ')
354 | disp( 'Pressione qualquer tecla para continuar... ')
355 | pause()
356 |
357 | s(2)
358 |
359 | disp( ' ')
360 | disp( 'Pressione qualquer tecla para continuar... ')
361 | pause()
362 |
363 | s(3)
364 |
365 | disp( ' ')
366 | disp( 'Pressione qualquer tecla para continuar... ')
367 | pause()
368 |
369 | s(1).nome
370 |
371 | disp( ' ')
372 | disp( 'Pressione qualquer tecla para continuar... ')
373 | pause()
374 |
375 | s(2).notas(2)
376 |
377 | disp( ' ')
378 | disp( 'Pressione qualquer tecla para continuar... ')
379 | pause()
380 |
381 | s(3).situacao
382 |
383 | disp( ' ')
384 | disp( 'Pressione qualquer tecla para continuar... ')
385 | pause()
386 |
387 | %
388 | %
389 | % EOF
390 | %
```

Código 1.3: Exemplos de operações a estruturas de dados.

```

2 %%%%
3 %
4 % Arquivo: Aula_2_Lab_Octave_Operacoes.m
5 %
6 %%%%
8 %
9 %
10 % Titulo:
11 % Exemplos de operações com estruturas de dados %
12 %
13 %
14 % Autor : Alexandre Santos de la Vega %
15 % Datas : /2022-01/
16 %
17 %%%%
18
20 % limpeza do ambiente de trabalho
21 %
22 clear all
23 close all
24
26 disp( ' ' )
27 disp( ' ' )
28 disp( ' ' ) =====)
29 disp( ' ' ) Exemplos de operações com estruturas de dados')
30 disp( ' ' ) =====)
31 disp( ' ' )
32
34 %
35 % Exemplos de escalar (matriz 1x1)
36 %
38 e = ( -4 * (3^2) ) + ( 6 * (125^(1/3)) )
40 e = ( -4 * power(3,2) ) + ( 6 * nthroot(125,3) )
42 %
44 % Exemplos de vetor linha (matriz 1xC)
45 %
46 vl = [ 1 2 3 ]
48 emvl = 4 * vl
50 vldde = vl / 2
52 vldee = 2 \ vl
54 vlmvl = vl .* vl
56 %
57 % Erro !!!

```

```

58  %
59  % vlmvl = vl * vl
60  vlddvl = vl ./ vl
61  vldevl = vl .\ vl
62
63  vlee = vl .^ 2
64
65  vlevl = vl .^ vl
66
67
68
69  %
70  % Exemplos de vetor coluna (matriz Lx1)
71  %
72
73  vc = [ 1 2 3 ]. '
74
75  emvc = 4 * vc
76
77  vcdde = vc / 2
78
79  vcdee = 2 \ vc
80
81  vcmvc = vc .* vc
82
83  %
84  % Erro !!!
85  %
86  % vcmvc = vc * vc
87
88  vcddvc = vc ./ vc
89
90  vcdevc = vc .\ vc
91
92  vcee = vc .^ 2
93
94  vcevc = vc .^ vc
95
96  %
97  % Exemplos de vetor linha e vetor coluna
98  %
99
100  vl = [ 1 3 5 ]
101
102  vc = [ 6 ; 4 ; 2]
103
104  vlpevc = vl * vc
105
106  %
107  % Erro !!!
108  %
109  % vlmvc = vl .* vc
110
111  vlx = [ 1+j 3+j*3 5+j*5 ]
112
113  vlxtc = vlx '
114
115  vlxt = vlx .
116

```

```

118  vcx = [ 2+j*2 ; 4+j*4 ; 6+j*6 ]
119  vcxtc = vcx ,
120  vcxt  = vcx ,
121
122
123
124 %
125 % Exemplos de vetor linha/coluna e matriz retangular (matriz LxC)
126 %
127
128  vl = [ 1 3 ]
129
130  vc = [ 6 ; 4 ; 2]
131
132  mr = [ 10 20 30 ; 40 50 60 ]
133
134  vlmmr = vl * mr
135
136  mrmvc = mr * vc
137
138
139 %
140 % Exemplos de matriz retangular (matriz LxC)
141 %
142
143
144  mr = [ 1 2 3 ; 4 5 6 ]
145
146  mrmmr = mr .* mr
147
148  mree = mr .^ 3
149
150  mrmmr = mr * mr.'
151
152  mrtmmr = mr.' * mr
153
154  mrx = [ 1 2+j*2 ; j*3 -4+j*4 ;
155          -5 -6-j*6 ; -j*7 8-j*8 ]
156
157  mrxj = conj(mrx)
158
159  real(mrx)
160
161  imag(mrx)
162
163  abs(mrx)
164
165  angle(mrx)
166
167  angle(mrx)*180/pi
168
169 %
170 % EOF
171 %

```

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 | %%>>> x = [0:pi/100:2*pi];
2 | %>>> y = sin(x);
3 | %>>> plot(x,y)
4 | %>>> title('Gráfico de sin(x)')
5 | %>>> xlabel('x')
6 | %>>> ylabel('y')
```

```

5 %%%%%%
7 %%%%%%
9 %%%%%%
% Titulo : %
11 % Exemplo de gráfico discreto de função unidimensional %
% %
13 % %
% Autor : Alexandre Santos de la Vega %
15 % Datas : /2022-01/
% %
17 %%%%%%
19 % limpeza do ambiente de trabalho
21 clear all
close all
23
% definicao da abscissa
25 n = -5:5;
27
% definicao da ordenada
a2 = 1;
29 a1 = 0;
a0 = 0;
31 y = a2.* (n.^2) + a1.*n + a0;
33
% criacao de uma tela de desenho (canvas)
figure (1)
35
% criacao de desenho
37 stem(n,y)
39
% criacao de labels
xlabel( 'n')
41 ylabel( 'y[n] = a_2 n^2 + a_1 n + a_0')
43
% criacao de titulo
title( 'Gráfico discreto de função unidimensional')
45
% controle de faixas
47 faixa = [-10 10 -5 30];
axis(faixa)
49
%
51 %EOF
%

```

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

```

2 %%%%%%
3 %
4 % Arquivo: Aula_3_Lab_Octave_Curvas_2.m
5 %
6 %%%%%%
7 %
8 %%%%%%
9 %
10 % Titulo: %
11 % Exemplo de criação de subjanelas em uma mesma tela %
12 %
13 %
14 % Autor : Alexandre Santos de la Vega %
15 % Datas : /2022-01/
16 %
17 %%%%%%
18 %
19 %
20 % limpeza do ambiente de trabalho
21 clear all
22 close all
23 %
24 % definicao da abscissa
25 n = -1.5:0.01:1.5;
26 %
27 % definicao da ordenada
28 y1 = (n);
29 y2 = (n.^2);
30 y3 = (n.^3);
31 y4 = (n.^4);
32 %
33 % criacao de uma tela de desenho (canvas)
34 figure (2)
35 %
36 % criacao de subjanela na tela
37 subplot(2,2,1)
38 %
39 % criacao de desenho
40 stem(n,y1,'b')
41 %
42 % criacao de titulo
43 title('Polinômios de grau ímpar')
44 %
45 % criacao de labels
46 ylabel('y[n] = n^1')
47 %
48 % criacao de subjanela na tela
49 subplot(2,2,2)
50 %
51 % criacao de desenho
52 stem(n,y2,'g')
53 %
54 % criacao de titulo
55 title('Polinômios de grau ímpar')
56 %
57 % criacao de labels
58 
```

```

58 | ylabel('y[n] = n^2')
60 | % criação de subjanelas na tela
61 | subplot(2,2,3)
62 |
63 | % criação de desenho
64 | stem(n,y3,'r')
65 |
66 | % criação de labels
67 | ylabel('y[n] = n^3')
68 | xlabel('n')
69 |
70 | % criação de subjanelas na tela
71 | subplot(2,2,4)
72 |
73 | % criação de desenho
74 | stem(n,y4,'k')
75 |
76 | % criação de labels
77 | ylabel('y[n] = n^4')
78 | xlabel('n')
79 |
80 |
81 | %
82 | %EOF
83 | %

```

Código 1.6: Exemplo de superposição de gráficos discretos.

```

1 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 | %
3 | % Arquivo: Aula_3_Lab_Octave_Curvas_3.m
4 | %
5 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 |
7 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8 | %
9 | % Titulo: %
10 | % Exemplo de superposição de gráficos discretos %
11 | %
12 | %
13 | %
14 | % Autor : Alexandre Santos de la Vega %
15 | % Datas : /2022-01/ %
16 | %
17 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
18 |
19 | % limpeza do ambiente de trabalho
20 | clear all
21 | close all
22 |
23 | % definição da abscissa
24 | n = -1.5:0.01:1.5;
25 |
26 | % definição da ordenada
27 | y1 = (n);

```

```

29 | y2 = (n.^2);
| y3 = (n.^3);
31 | y4 = (n.^4);

33 | % criação de uma tela de desenho (canvas)
| figure (3)
35 |
36 | % habilitação da superposição
37 | hold on

39 | % criação de desenho
| stem(n,y1,'b')
41 | stem(n,y2,'g')
| stem(n,y3,'r')
43 | stem(n,y4,'k')

45 | % criação de legenda
| legend('n^1','n^2','n^3','n^4', 'Location', 'southeast')
46 |
47 | % criação de labels
49 | xlabel('n')
| ylabel('y[n] = n^k , k = 1:4')

51 |
52 | % criação de título
53 | title('Superposição de gráficos discretos')

55 |
56 | %EOF
57 |

```

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

```

1 | %%%%%%
| %
3 | % Arquivo: Aula_3_Lab_Octave_Curvas_4.m
| %
5 | %%%%%%
7 |
9 | %
| %
% Titulo:
| %
11 | % Exemplo de criação de subjanela ocupando vários espaços %
| %
13 | %
| %
% Autor : Alexandre Santos de la Vega %
15 | % Datas : /2022-01/
| %
17 | %%%%%%
19 |
| %
% limpeza do ambiente de trabalho
21 | clear all
| close all
23 |
| %
% definição da abscissa
25 | 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

```

```

85 | hold on
87 | % criação de desenho na tela
88 | stem(n,y1,'b')
89 | stem(n,y2,'g')
90 | stem(n,y3,'r')
91 | stem(n,y4,'k')
92 |
93 | % criação de legenda
94 | legend('n^1','n^2','n^3','n^4','Location','southeast')
95 |
96 | % criação de labels
97 | xlabel('n')
98 | ylabel('y[n] = n^k', 'k = 1:4')
99 |
100 | % criação de título
101 | title('Superposição de gráficos discretos')
102 |
103 |
104 | %
105 | %EOF
106 | %

```

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

```

2 | %%%%%%
2 | %
2 | % Arquivo: Aula_3_Lab_Octave_Curvas_5.m
4 | %
4 | %%%%%%
6 |
8 | %%%%%%
8 | %
8 | %%%%%%
10 | % Titulo:
10 | %
10 | % Exemplo de criação:
12 | %
12 | % de curva discreta,
12 | %
12 | % de curva contínua por meio de interpolador de ordem 0 %
14 | %
14 | % e
14 | %
14 | % de curva contínua por meio de interpolador de ordem 1 %
16 | %
16 | %
18 | % Autor : Alexandre Santos de la Vega
18 | %
18 | % Datas : /2022-01/
20 | %
20 | %%%%%%
22 |
24 | % limpeza do ambiente de trabalho
24 | clear all
26 | close all
28 | % definição do período
28 | Np = 40;
30 | %
30 | % definição do índice
32 | 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
%
62 subplot(2,2,2)
stairs(t,y,'r')
title( {'Curva produzida por interpolador de ordem 0 (ZOH)', ...
        'com Fs = 44 kHz' })
64 ylabel('y(t) = ZOH \{ y[n] \}')
xlabel('t (s)')
axis([0 t(end) min(y) max(y)])
66
%
68 subplot(2,2,3)
plot(t,y,'b')
title( {'Curva produzida por interpolador de ordem 1 (FOH)', ...
        'com Fs = 44 kHz' })
70 ylabel('y(t) = FOH \{ y[n] \}')
xlabel('t (s)')
axis([0 t(end) min(y) max(y)])
72
%
74 subplot(2,2,4)
hold on
76
78 stem(t,y,'k')
80 stairs(t,y,'r')
plot(t,y,'b')
82 title('Superposição das curvas')
84 ylabel('y[n] = sin (\Omega n)')
86 xlabel('t (s)')
axis([0 t(end) min(y) max(y)])
88
%
90 %EOF
%
```

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.

```

2 |%%%%%
2 |%
2 |% Arquivo original : Aula_4_Lab_Octave_Curva_3D.m
4 |% Arquivo atualizado : Exp_sin_cos_2021_07_05.m
4 |%
6 |% Arquivo : Aula_4_Lab_Octave_Exp_3D_Sin_Cos.m
8 |%%%%%

```

```

10 %
12 %
% Titulo :
14 % Exemplo de criacao de curva 3D %
% - Relacoes: exp(), sin() e cos(). %
16 % - Casos: continuo e discreto. %
18 %
% Autor : Alexandre Santos de la Vega %
20 % Datas : /2021_07_05/
% %
22 %

24 %
% limpeza do ambiente de trabalho
26 clear all
close all
28 %
% definicao do periodo
30 Np = 40;
32 %
% definicao do indice
n = 0:Np;
34 %
% definicao da frequencia discreta
36 Omega = 2*pi / Np;
38 %
% definicao da abscissa
x = cos(Omega * n);
40 %
% definicao da ordenada
42 y = sin(Omega * n);
44 %
% definicao da altura
z = n;
46 %
% definicao do periodo de amostragem e do tempo
48 Ts = 1 / 44e3;
t = n * Ts;
50

52 %
FigNbr = 0;
54 black_val = 0;

56 %
FigNbr = FigNbr + 1;
58 figure(FigNbr)
%
60 %
subplot(2,2,1)
62 plot(x,y, "o")
axis square
64 title('Curva discreta 2D: e^{j (\Omega n)}')
66 xlabel('y[n] = sin (\Omega n)')
68 xlabel('x[n] = cos (\Omega n)')
%
68 % fundo preto

```

```

70  %set(gcf, 'color',[black_val, black_val, black_val])
71  %
72  subplot(2,2,2)
73  stem(n,y)
74  axis([0 n(end) min(y) max(y)])
75  title('Projecao no eixo Y')
76  ylabel('y[n] = sin (\Omega n)')
77  xlabel('n')
78  % fundo preto
79  %set(gcf, 'color',[black_val, black_val, black_val])
80  %
81  subplot(2,2,3)
82  stem(n,x)
83  axis square
84  axis([0 n(end) min(y) max(y)])
85  title('Projecao no eixo X')
86  ylabel('x[n] = cos (\Omega n)')
87  xlabel('n')
88  % fundo preto
89  %set(gcf, 'color',[black_val, black_val, black_val])
90  %
91  %view (AZIMUTH, ELEVATION)
92  view(90,90)

93  %
94  disp(' ')
95  disp('Pressione qualquer tecla para continuar... ')
96  pause()
97  %

98  %
99  FigNbr = FigNbr + 1;
100 figure(FigNbr)
101 %
102 subplot(2,2,1)
103 plot(x,y)
104 axis square
105 title('Curva continua 2D: e^{j (\omega t)}')
106 ylabel('y(t) = sin (\omega t)')
107 xlabel('x(t) = cos (\omega t)')
108 %

109 %
110 % fundo preto
111 %set(gcf, 'color',[black_val, black_val, black_val])
112 %
113 subplot(2,2,2)
114 plot(t,y)
115 axis([0 t(end) min(y) max(y)])
116 title('Projecao no eixo Y')
117 ylabel('y(t) = sin (\omega t)')
118 xlabel('t')
119 %

120 %
121 %fund preto
122 %set(gcf, 'color',[black_val, black_val, black_val])
123 %
124 subplot(2,2,3)
125 plot(t,x)
126 axis([0 t(end) min(y) max(y)])
127 title('Projecao no eixo X')

```

```

128 | ylabel('x(t) = cos (\omega t)')
129 | xlabel('t')
130 | %
131 | % fundo preto
132 | %set(gca, 'color',[black_val, black_val, black_val])
133 | %
134 | %view (AZIMUTH, ELEVATION)
135 | view (90,90)
136 |
137 | %
138 | disp(' ')
139 | disp('Pressione qualquer tecla para continuar... ')
140 | pause()
141 |
142 | %
143 | FigNbr = FigNbr + 1;
144 | figure(FigNbr)
145 | %
146 | subplot(2,2,1)
147 | plot(x,y, "o")
148 | axis square
149 | title('Projecao no plano X x Y')
150 | ylabel('y[n] = sin (\Omega n)')
151 | xlabel('x[n] = cos (\Omega n)')
152 |
153 | % fundo preto
154 | %set(gca, 'color',[black_val, black_val, black_val])
155 | %
156 | subplot(2,2,2)
157 | stem(n,y)
158 | axis([0 n(end) min(y) max(y)])
159 | title('Projecao no plano Z x Y')
160 | ylabel('y[n] = sin (\Omega n)')
161 | xlabel('z[n] = n')
162 |
163 | % fundo preto
164 | %set(gca, 'color',[black_val, black_val, black_val])
165 | %
166 | subplot(2,2,3)
167 | stem(n,x)
168 | axis square
169 | axis([0 n(end) min(y) max(y)])
170 | title('Projecao no plano Z x X')
171 | ylabel('x[n] = cos (\Omega n)')
172 | xlabel('z[n] = n')
173 |
174 | % fundo preto
175 | %set(gca, 'color',[black_val, black_val, black_val])
176 | %
177 | %view (AZIMUTH, ELEVATION)
178 | view (90,90)
179 |
180 | subplot(2,2,4)
181 | stem3(x,y,z)
182 | title('Curva discreta 3D: e^{j (\Omega n)}')
183 | zlabel('z[n] = n')
184 | ylabel('y[n] = sin (\Omega n)')
185 | xlabel('x[n] = cos (\Omega n)')
186 | %

```

```

188 % fundo preto
189 %set(gca, 'color',[black_val, black_val, black_val])
190 %
191 disp(' ')
192 disp('Pressione qualquer tecla para continuar... ')
193 pause()
194 %
195 FigNbr = FigNbr + 1;
196 figure(FigNbr)
197 %
198 subplot(2,2,1)
199 plot(x,y)
200 axis square
201 title('Projecao no plano X x Y')
202 ylabel('y(t) = sin (\omega t)')
203 xlabel('x(t) = cos (\omega t)')
204 %
205 % fundo preto
206 %set(gca, 'color',[black_val, black_val, black_val])
207 %
208 subplot(2,2,2)
209 plot(t,y)
210 axis([0 t(end) min(y) max(y)])
211 title('Projecao no plano Z x Y')
212 ylabel('y(t) = sin (\omega t)')
213 xlabel('z(t) = t')
214 %
215 % fundo preto
216 %set(gca, 'color',[black_val, black_val, black_val])
217 %
218 subplot(2,2,3)
219 plot(t,x)
220 axis square
221 axis([0 t(end) min(y) max(y)])
222 title('Projecao no plano Z x X')
223 ylabel('x(t) = cos (\omega t)')
224 xlabel('z(t) = t')
225 %
226 % fundo preto
227 %set(gca, 'color',[black_val, black_val, black_val])
228 %
229 %view (AZIMUTH, ELEVATION)
230 view(90,90)
231 %
232 subplot(2,2,4)
233 plot3(x,y,t)
234 grid on
235 title('Curva continua 3D: e^{j (\omega t)}')
236 zlabel('z(t) = t')
237 ylabel('y(t) = sin (\omega t)')
238 xlabel('x(t) = cos (\omega t)')
239 %
240 % fundo preto
241 %set(gca, 'color',[black_val, black_val, black_val])
242 %
243 %
244 %EOF

```

246 | %

Código 1.10: Exemplo de criação de superfície discreta.

```

2 %
% Arquivo: Aula_4_Lab_Octave_Superficies.m
4 %
%
6 %

8 %
% %
10 % Titulo:
% Exemplo de criacao de superficie discreta 3D %
12 %
%
14 % Autor : Alexandre Santos de la Vega %
15 % Datas : /2022-01/
16 %
%
18 %

20 % limpeza do ambiente de trabalho
clear all
21 close all

24 %
x = -1:0.01:1;
25 y = x;
[X,Y] = meshgrid(x,y);
26 %
%
27 %z = sqrt(X) + sqrt(Y);
28 z = X.^2 + Y.^2;
29 %
30 %
31 FigNbr = 0;

32 %
33 FigNbr = FigNbr + 1;
34 figure(FigNbr)
35 %
36 surf(X,Y,z)

37 disp(' ')
38 disp('Pressione qualquer tecla para continuar... ')
39 pause()

40 %
41 FigNbr = FigNbr + 1;
42 figure(FigNbr)
43 %
44 surf(X,Y,z)

45 %
46 disp(' ')
47 disp('Pressione qualquer tecla para continuar... ')
48 pause()

```

```
54 | pause()
56 | %
57 | FigNbr = FigNbr + 1;
58 | figure(FigNbr)
59 | %
60 | mesh(X,Y,z)

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

66 | %
67 | FigNbr = FigNbr + 1;
68 | figure(FigNbr)
69 | %
70 | meshc(X,Y,z)

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

76 | %
77 | FigNbr = FigNbr + 1;
78 | figure(FigNbr)
79 | %
80 | meshz(X,Y,z)

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

86 | %
87 | FigNbr = FigNbr + 1;
88 | figure(FigNbr)
89 | %
90 | waterfall(X,Y,z)

92 | %
93 | % 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

Imagen indexada

- Utiliza duas matrizes por imagem: matriz 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 → double: $M64 = \text{double}(M8) + 1$.
 - * uint16 → double: $M64 = \text{double}(M16) + 1$.
 - * double → uint8 : $M8 = \text{uint8}(\text{round}(M64 - 1))$.
 - * double → uint16: $M16 = \text{uint16}(\text{round}(M64 - 1))$.
- 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.

Imagen 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].

Imagen RGB ou *Truecolor*

- Utiliza uma matriz por imagem: matriz de *pixels*.
- Matriz de *pixels* ($L \times C \times 3$): 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, colordcube, 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.

```

2 %%%%%
3 %
4 % Arquivo : Aula_5_Lab_Octave_Imagens_Basics.m
5 %
6 %
7 %%%%%
8 %
9 %
10 % Titulo :
11 % Exemplo de gráfico discreto de função bidimensional %
12 % - Imagem %
13 %
14 % Exemplos básicos na geração de imagens em tons de cinza %
15 %
16 %
17 % Autor : Alexandre Santos de la Vega %
18 % Datas : /2022-01/
19 %
20 %
21 %
22 %
23 %
24 =====
25 %
26 % limpeza do ambiente de trabalho %
27 %
28 clear all

```

```

30 | close all
32 | %
33 | % _____
34 | %
35 |
36 | % define niveis de branco e preto
37 | c_black = 0;
38 | c_white = 64;
39 |
40 | % inicializa o numero do canvas
41 | FigNbr = 0;
42 |
43 | %
44 | % _____
45 | %
46 |
47 | % desenha 1 pixel
48 | %
49 | FigNbr = FigNbr+1 ;
50 | figure(FigNbr)
51 | %
52 | map = gray;
53 | colormap(map);
54 | %
55 | m_image = [c_black];
56 | %
57 | image(m_image)
58 | f = size(m_image);
59 | axis([0 f(2)+1 0 f(1)+1])
60 |
61 | disp(' ')
62 | disp('Pressione qualquer tecla para continuar... ')
63 | pause()
64 |
65 | axis square
66 | %axis image
67 |
68 | disp(' ')
69 | disp('Pressione qualquer tecla para continuar... ')
70 | pause()
71 |
72 | %
73 | % _____
74 | %
75 |
76 | % desenha alguns pixels na forma de uma matriz
77 | %
78 | FigNbr = FigNbr+1 ;
79 | figure(FigNbr)
80 | %
81 | map = gray;
82 | colormap(map);
83 | %
84 | m_image = c_white * ones(3,3);
85 | m_image([1,3],[1,3]) = c_black;
86 | %
87 | image(m_image)
88 | f = size(m_image);

```

```
90 | axis([0 f(2)+1 0 f(1)+1])
90 |
91 | disp(' ')
92 | disp('Pressione qualquer tecla para continuar... ')
92 | pause()
94 |
94 | axis square
96 | %axis image
98 |
98 | disp(' ')
98 | disp('Pressione qualquer tecla para continuar... ')
98 | pause()
100 |
102 |
102 | %
104 |
104 | %
106 | % desenha alguns pixels na forma de uma matriz
106 | %
108 | FigNbr = FigNbr+1 ;
108 | figure(FigNbr)
110 | %
110 | map = gray;
112 | colormap(map);
112 | %
114 | m_image = c_white * ones(5,5);
114 | m_image(1:2:5,1:2:5) = c_black;
116 | %
116 | image(m_image)
118 | f = size(m_image);
118 | axis([0 f(2)+1 0 f(1)+1])
120 |
120 | disp(' ')
122 | disp('Pressione qualquer tecla para continuar... ')
122 | pause()
124 |
124 | axis square
126 | %axis image
128 |
128 | disp(' ')
128 | disp('Pressione qualquer tecla para continuar... ')
130 | pause()
132 |
132 | %
134 |
134 | %
136 | % desenha uma cruz
136 | %
138 | FigNbr = FigNbr+1 ;
138 | figure(FigNbr)
140 | %
140 | map = gray;
142 | colormap(map);
142 | %
144 | m_image = c_white * ones(3,3);
144 | m_image(1:2:5,1:2:5) = c_black;
146 | %
146 | image(m_image)
```

```

148 f = size(m_image);
149 axis([0 f(2)+1 0 f(1)+1])
150
151 disp(' ')
152 disp('Pressione qualquer tecla para continuar... ')
153 pause()
154
155 axis square
156 %axis image
157
158 disp(' ')
159 disp('Pressione qualquer tecla para continuar... ')
160 pause()
161
162 %
163 %=====
164 %

165 % desenha sinais de + e - dentro de um frame
166 %
167 FigNbr = FigNbr+1 ;
168 figure(FigNbr)
169 %
170 map = gray;
171 colormap(map);
172 %
173 m_image = c_white * ones(15,15);
174 %
175 m_image(1 , 1:end) = c_black;
176 m_image(1:end , end ) = c_black;
177 m_image(end , 1:end) = c_black;
178 m_image(1:end , 1 ) = c_black;
179 %
180 m_image(1+3 , 1+(2:4)) = c_black;
181 m_image(1+(2:4) , 1+3 ) = c_black;
182 %
183 m_image(end-3 , end-(2:4)) = c_black;
184 %
185 image(m_image)
186 f = size(m_image);
187 axis([0 f(2)+1 0 f(1)+1])
188
189 disp(' ')
190 disp('Pressione qualquer tecla para continuar... ')
191 pause()
192
193 axis square
194 %axis image
195
196 disp(' ')
197 disp('Pressione qualquer tecla para continuar... ')
198 pause()
199
200 %
201 %EOF
202 %

```

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

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %
3 % Arquivo: heatmap_propagacao_modelo_de_friis.m.m
4 %
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 %
7 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8 %
9 % Arquivo: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
10 % - Heatmap, com definição de cMap. %
11 % - Modelo de espaço livre de Friis. %
12 % - Gráfico de Perda. %
13 % - Identificação de posição de TX. %
14 % - Identificação de campo distante. %
15 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16 %
17 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
18 % Autor: Alexandre Santos de la Vega %
19 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
20 % Data: /2024-01/
21 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
22 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
23 %
24 %=====
25 %
26 %
27 % limpeza do ambiente de trabalho
28 %
29 clear all
30 close all
31 %
32 %
33 %=====
34 %
35 %
36 %
37 % define parâmetros comuns
38 %
39 % velocidade da luz no vácuo
40 c = 3e8 ; % m/s
41 %
42 % frequência da antena TX
43 Ft = 1e6 ; % Hz
44 %
45 % comprimento de onda
46 lambda = (c / Ft) ; % m
47 %
48 % resolução da amostragem
49 y_res = .25 * lambda ; % m
50 x_res = .25 * lambda ; % m
51 %
52 %
53 %=====
54 %
55 % define ambiente de propagação
56 %
57 
```

```

59 % limite contínuo
y_lim = 8e3 ; % max y (m)
x_lim = 12e3 ; % max x (m)
61
% limite discreto
63 r_size = fix(y_lim / y_res) + 1 ; % max linha
c_size = fix(x_lim / x_res) + 1 ; % max coluna
65
% matriz de distâncias em relação à antena TX
67 m_dist = zeros(r_size , c_size) ; % ambiente discreto

69 %
% =====
71 %

73 % define posição da antena TX

75 % posição contínua
%
77 % for testing ...
y_tx_pos = y_lim/2 ; % m
79 x_tx_pos = x_lim/2 ; % m
%
81 % desired ...
%y_tx_pos = 10 ; % m
83 %x_tx_pos = 20 ; % m

85 % posição discreta
r_tx_pos = fix(y_tx_pos / y_res) ;
c_tx_pos = fix(x_tx_pos / x_res) ;

89 %
% =====
91 %

93 % realiza cálculo das distâncias
%
95 for row=1:r_size
    for col=1:c_size
        m_dist (row,col) = sqrt ( ( (row*y_res) - y_tx_pos )^2
97                                +
99                                ( (col*x_res) - x_tx_pos )^2 ) ;
    endfor
101 endfor
%
103 clear row col;

105 %%%%%%
107 % realiza cálculo das variáveis (potência , perda , etc .)

109 % define parâmetros
Pt = .25 ; % W
111 Gt = 1 ;
Gr = 1 ;
113
% calcula fator de perda
115 mAtr = ( lambda ./ (4 * pi * m_dist) ) .^ 2 ;

```

```

117 % calcula a potência recebida
%
119 % Modelo de Free Space de Friis
Pr = Pt * Gt * Gr * mAttr ;
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 visualização
%
173 if (axis_choice==0)
%   limites discretos
175 x_lims = [0 , (c_size-1)] ;

```

```

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

183 %%%%%%
185 % desenha a imagem da matriz no canvas ,
185 % usando o mapa de cores escolhido .
187 %
187 function_choice = 1 ; % imagesc()
189 %function_choice = 2 ; % image()
189 %function_choice = 3 ; $ imshow()
191 %
191 if (function_choice==1)
193     imagesc(x_lims,y_lims,z) ;
193 elseif (function_choice==2)
195     image(x_lims,y_lims,z) ;
195 else
197     % imshow(z,"colormap",cMap,"xdata",x_lims,"ydata",y_lims);
197     imshow(z,cMap) ;
199 endif
199 %
201 % retira a ordem "reverse" do eixo y...
201 set(gca,"YDir","normal") ;
203 %
205 % define aspect ratio
205 daspect ([1 1])

207 %%%%%%
209 % coloca identificadores
209 %
211 tit_str = cellstr(
211     ['Heatmap do modelo de espaço livre de Friis';
211     'Gráfico da perda: L = (Pt / Pr)';
211     'Identificações: posição do TX e início do campo distante.';
211     'Parâmetros: c = ', num2str(c), ...
211     ' m/s ; f = ', num2str(Ft), ...
211     ' Hz ; \lambda = ', num2str(lambda), ' m ; ';
211     'Pt = ', num2str(Pt), ...
211     ' W ; Gt = ', num2str(Gt), ...
211     ' ; Gr = ', num2str(Gr), ' .';
211     'Resoluções: x = ', num2str(x_res), ...
211     ' m ; y = ', num2str(y_res), ' m .']);
223 title(tit_str)
223 %
225 if (axis_choice==0)
225     % limites discretos
225 %
227     ylabel('Linha')
229 xlabel('Coluna')
231 else
231     % limites contínuos
231 %
233     ylabel('Dimensão y (m)')
233 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 antenna TX
%
255 | if (axis_choice==0)
% limites discretos
%
257 | 
258 |   plot(c_tx_pos , r_tx_pos , "k*")
259 | else
% limites contínuos
%
261 | 
262 |   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*")
266 | endif
267 |
268 | %%%%%%
269 | % desenha um círculo
270 | % no inicio de campo distante de Friis
%
273 | if (axis_choice==0)
% limites discretos
%
275 | 
276 |   r_disc = 10*lambda;
277 |   theta = 0:(pi/360):(2*pi);
%
279 |   c_cd = c_tx_pos + fix(r_disc / x_res) * cos(theta) ;
280 |   r_cd = r_tx_pos + fix(r_disc / y_res) * sin(theta) ;
%
281 |   plot(c_cd , r_cd , "k")
283 | else
% limites contínuos
%
285 | 
286 |   r_cont = 10*lambda;
287 |   theta = 0:(pi/360):(2*pi);
%
289 |   x_cd = x_tx_pos + r_cont * cos(theta) ;
290 |   y_cd = y_tx_pos + r_cont * sin(theta) ;
%
291 |   plot(x_cd , y_cd , "k")
293 | endif

```

```

295 | %%%%%%
297 | %
298 | % =====
299 | %
300 | %
301 | %
302 | % 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

$$a_0y[n] + a_1y[n - 1] + \dots + a_Ny[n - N] = b_0x[n] + b_1x[n - 1] + \dots + b_Lx[n - L] . \quad (1.1)$$

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

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

- 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

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

dados os vetores

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

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

e

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

- 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.

```

1 %%%%%%%%%%%%%%
2 %
3 % Arquivo: Aula_7_Lab_Octave_ED_e_filter.m
4 %
5 %%%%%%%%%%%%%%
6 %
7 %%%%%%%%%%%%%%
8 %
9 % Titulo:
10 %%%%%%%%%%%%%%
11 % Exemplo de simulação de uma equação de diferença %
12 %%%%%%%%%%%%%%
13 %
14 %%%%%%%%%%%%%%
15 % Autor : Alexandre Santos de la Vega %
16 %%%%%%%%%%%%%%
17 %%%%%%%%%%%%%%
18 %
19 %
20 %=====
21 %
22 %
23 % limpeza do ambiente de trabalho
24 %
25 clear all
26 close all
27 %
28 %
29 %
30 %=====
31 %
32 %
33 % define os coeficientes da equacao de diferenca
34 %
35 % b = [ b0 b1 b2 ]
36 b = [ 0.435390588649551 0.507111243808470 0.435390588649552 ];
37 %
38 % a = [ a0 a1 a2]
39 a = [ 1.000000000000000 0.113933557377612 0.263958863729961 ];
40 %
41 % inicializa o numero do canvas
42 FigNbr = 0;
43 %
44 %=====
45 %
46 %
47 % define um impulso discreto unitario
48 %
49 Nud = 21;
50 %
51 unit_delta = zeros(1,Nud);
52 unit_delta(1) = 1;
53 
```

```

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)
%
%
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]')
title('Degrau discreto unitário')
111 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 |
131 | xlabel( 'n' )
132 |
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 |
144 | subplot(3,2,4)
145 | stem( (0:(Nus-1)) , y_us )
146 | 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 |
166 | xlabel( 'n' )
167 |
168 | %EOF
169 | %

```

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:

$$m_1 = \begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix}, m_2 = \begin{bmatrix} 10 & 9 \\ 14 & 13 \end{bmatrix}, m_3 = \begin{bmatrix} 7 & 8 \\ 3 & 4 \end{bmatrix} \text{ e } m_4 = \begin{bmatrix} 11 & 15 \\ 12 & 16 \end{bmatrix}.$$

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:

$$v_1 = [1, 7, 31, 22, 93], v_2 = [88, -31, 108, 14, 62], v_3 = [72, -11], m_4 = \begin{bmatrix} 16 & 25 \\ 48 & 53 \end{bmatrix} \text{ e } m_5 = \begin{bmatrix} 0.3 & 6 \\ 1 & 19 \end{bmatrix}.$$

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:

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

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 \mathbf{D}_N . Em seguida, para $W_N = e^{-j(\frac{2\pi}{N})}$ e um dado vetor $\mathbf{x}[n] = [x[0] \ x[1] \ \cdots \ x[N-1]]$, calcule matriz \mathbf{D}_N e o vetor $\mathbf{X}[k] = [X[0] \ X[1] \ \cdots \ X[N-1]]$ usando a equação matricial proposta.

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

$$\mathbf{X}[k] = \mathbf{D}_N \ \mathbf{x}[n], \quad 0 \leq (k, n) \leq (N-1) \quad (1.4)$$

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*.

```
2 %  
3 % Arquivo: quant_half_grid.m  
4 %  
5 %  
6 %function xq = quant_half_grid(x, min_grid, grid_step, max_grid)  
7 %  
8 % QUANT_HALF_GRID quantization with half grid trigger.  
9 %  
10%  
11 xq = x;  
12 half_grid = grid_step / 2;  
13 for k = 1:length(x)  
14     for grid_val = min_grid:grid_step:max_grid  
15         dist = x(k) - grid_val;  
16         if ( abs(dist) < half_grid ) ...  
17             || ...  
18             ( (dist < 0) && (abs(dist) == half_grid) ) ...  
19         )  
20             xq(k) = grid_val;
```

```

22      break
      end
      end
24 end
%
26 % EOF
%

```

Código 2.2: Exemplos de sinais de diferentes tipos.

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %
3 % Arquivo: tipos_de_sinais.m
4 %
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7 %
8 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9 %
10 % Demos para
11 % Apostila de DSP
12 %
13 % Topico: tipos de sinais
14 %
15 % Autor : Alexandre Santos de la Vega
16 % Modifs: 2022-1/2021-2/2011-02/2010-01/
17 %
18 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
19 %
20 %
21
22 % limpa ambiente
23 clear all
24 close all
25
26
27 % define parametros
28 A0 = 1;
29 F0 = 100;
30 T0 = 1/F0;
31 %
32 % numero de periodos visualizados
33 NOP = 2;
34 %
35 % numero de pontos por periodo para sinal analogico
36 PPPa = 1000;
37 Tsa = T0/PPPa;
38 t = 0:Tsa:(NOP*T0);
39 %
40 % numero de pontos por periodo para sinal amostrado
41 PPPs = 10;
42 Tss = T0/PPPs;
43 nTs = 0:Tss:(NOP*T0);
44 %
45 n = 0:(length(nTs) - 1);
46
47

```

```

49 % constroi sinais
50 xa = A0*cos(2*pi*F0*t); % sinal analogico
51 xs = A0*cos(2*pi*F0*nTs); % sinal amostrado
52
53 % quantiza sinais usando funcao definida externamente
54 %
55 % numero de intervalos do grid
56 NOI = 5;
57 % sinal analogico quantizado
58 xaq = quant_half_grid(xa,-A0,(A0/NOI),A0);
59 % sinal amostrado quantizado
60 xsq = quant_half_grid(xs,-A0,(A0/NOI),A0);
61
62 %
63 %
64 % desenha graficos
65 %
66 %
67 %
68 figure(1)
69 %
70 % black = 0.0 ; gray_suave = 0.8 ; white = 1.0
71 BCV = 1.0;
72 back_color = [BCV,BCV,BCV];
73
74 %
75 % sem quantizacao
76 %
77 subplot(3,3,1)
78 plot(t,xa)
79 ylabel('x(t)')
80 xlabel('t (s)')
81 %
82 set(gca,'Color',back_color)
83 title({'Sinal analógico: x(t)', ...
84     ['T_s= ', num2str(T0/PPPs), 's', ' ; ', ...
85     'Q_res= ', num2str(1/NOI)]})
86 %
87 subplot(3,3,4)
88 stem(nTs,xs)
89 ylabel('x(nT_s)')
90 xlabel('nT_s (s)')
91 title({'Sinal amostrado', 'x(nT_s) = x(t) ; t = nT_s'})
92 %
93 set(gca,'Color',back_color)
94 %
95 subplot(3,3,7)
96 stem(n,xs)
97 ylabel('x[n]')
98 xlabel('n')
99 %
100 set(gca,'Color',back_color)
101 title({'Sequência amostrada', 'x[n] = x(nT_s)'})
102 %
103 % com quantizacao
104 %
105 subplot(3,3,2)
106 plot(t,xa,'b', t,xaq,'r')

```

```

107 | ylabel( '\{x(t)\}_Q' )
108 | 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)
% 'funciona, mas as cores nao ficam ok! ... '
%stem(n,[xs ; xsq])
hold on
127 | stem(n,xs,'bo-')
128 | stem(n,xsq,'ro-')
129 | ylabel( '\{x[n]\}_Q' )
xlabel( 'n' )
130 | title({ 'Sequência digital' , ...
          '\{x[n]\}_Q = \{x(nT_s)\}_Q' })
%
132 | set(gca , 'Color' , back_color)
%
134 | % com quantizacao e normalizado
%
136 | subplot(3,3,3)
137 | plot(t,NOI*xa,'b', t,NOI*xaq,'r')
%
139 | v = axis;
v(3) = -NOI;
140 | v(4) = NOI;
axis(v)
%
142 | ylabel( '\{x(t)\}_{Qn}' )
143 | xlabel( 't (s)' )
144 | title({ 'Sinal quantizado normalizado' , ...
          '\{x(t)\}_{Qn}' })
%
146 | set(gca , 'Color' , back_color)
%
148 | subplot(3,3,6)
% 'funciona, mas as cores nao ficam ok! ... '
%stem(nTs,[xs ; xsq])
hold on
150 | stem(nTs,NOI*xs,'bo-')
151 | stem(nTs,NOI*xsq,'ro-')
%
153 | v = axis;
v(3) = -NOI;
154 | v(4) = NOI;
axis(v)
%
156 | %

```

```

167 | ylabel( '\{x(nT_s)\}_\{Qn\} ')
168 | xlabel( 'nT_s (s)')
169 | title({ 'Sinal digital normalizado' , ...
170 |           '\{x(nT_s)\}_\{Qn\} = \{x(t)\}_\{Qn\} ; t = nT_s' })
171 | %
172 | set(gca, 'Color', back_color)
173 | %
174 | subplot(3,3,9)
175 | % 'funciona, mas as cores nao ficam ok! ... '
176 | %stem(n,[xs ; xsq])
177 | hold on
178 | stem(n,NOI*xs, 'bo-')
179 | stem(n,NOI*xsq, 'ro-')
180 | %
181 | v = axis;
182 | v(3) = -NOI;
183 | v(4) = NOI;
184 | axis(v)
185 | %
186 | ylabel( '\{x[n]\}_\{Qn\} ')
187 | xlabel( 'n')
188 | title({ 'Sequência digital normalizada' , ...
189 |           '\{x[n]\}_\{Qn\} = \{x(nT_s)\}_\{Qn\}' })
190 | %
191 | set(gca, 'Color', back_color)
192 |
193 | %%

```

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.

```

2 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 | %
4 | % Arquivo: disc_seq_from_ana_sig.m
5 | %
6 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7 |
8 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9 | %
10 | % Exemplo sobre %
11 | % geracao de sequencia discreta %
12 | % a partir de sinal analogico %
13 | %
14 | % Autor : Alexandre S. de la Vega %
15 | % Versao: 2018_2 %
16 | %
17 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
18 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

20 %
21 clear all
22 close all
23 %
24 %%%%%%
25 %
26 %%%%%%
27 %
28 %
29 A0 = 1;
30 f0 = 1e3; % f0 = 1 kHz
31 phi0 = -pi/4;
32 %
33 %
34 N0 = 10;
35 Omega0 = 2*pi/N0;
36 %
37 %
38 Fs = N0 * f0 ;
39 %
40 %
41 NOP = 3; % Nbr Of Periods
42 %
43 %%%%%%
44 %
45 %
46 %%%%%%
47 %
48 %
49 NOPPP = 100; % Nbr Pts Per Period
50 %
51 %
52 t_min = 0;
53 t_step = (1/(f0*NOPPP));
54 t_max = (NOP/f0);
55 %
56 t = t_min:t_step:t_max;
57 %
58 xt = A0 * cos(2*pi*f0*t + phi0);
59 %
60 %%%%%%
61 %
62 %
63 n_min = 0;
64 n_max = (3*N0);
65 %
66 n = n_min:n_max;
67 %
68 xn = A0 * cos(Omega0*n + phi0);
69 %
70 %%%%%%
71 %
72 %
73 %
74 %
75 %
76 FigNbr = 0;

```

```

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

82 %%%%%%
84 %
85 FigNbr = FigNbr + 1;
86 hf=figure(FigNbr);
87 set(hf, "papertype", "a4")
88 %
89 plot(t, xt, 'k')
90 v(2) = t_max;
91 v(3) = -EfctAmp;
92 v(4) = EfctAmp;
93 axis(v);
94 title('Sinal analógico')
95 ylabel('x(t)')
96 xlabel('t (s)')
97 %

98 %%%%%%
100 %
101 %
102 FigNbr = FigNbr + 1;
103 hf=figure(FigNbr);
104 set(hf, "papertype", "a4")
105 %
106 plot(t, xt, 'k')
107 hold on
108 stem(n/Fs, xn, 'r')
109 %
110 v(2) = t_max;
111 v(3) = -EfctAmp;
112 v(4) = EfctAmp;
113 axis(v);
114 title('Sinal analógico e Sinal analógico amostrado')
115 ylabel('x(t) e x(nTs)')
116 xlabel('t (s)')
117 %

118 %%%%%%
120 %
121 %
122 FigNbr = FigNbr + 1;
123 hf=figure(FigNbr);
124 set(hf, "papertype", "a4")
125 %
126 stem(n/Fs, xn, 'r')
127 v(2) = t_max;
128 v(3) = -EfctAmp;
129 v(4) = EfctAmp;
130 axis(v);
131 title('Sinal analógico amostrado')
132 ylabel('x(nTs)')
133 xlabel('nTs (s)')
134 %

135 %%%%%%
136 %

```

```

138    %
139    FigNbr = FigNbr + 1;
140    hf=figure(FigNbr);
141    set(hf, "papertype", "a4")
142    %
143    stem(n, xn, 'b')
144    v(2) = n_max;
145    v(3) = -EfctAmp;
146    v(4) = EfctAmp;
147    axis(v);
148    title('Sinal discreto')
149    ylabel('x[n]')
150    xlabel('n')
151    %
152    %%%%%%%%%%%%%%
153    %
154    %
155    FigNbr = FigNbr + 1;
156    hf=figure(FigNbr);
157    set(hf, "papertype", "a4")
158    %
159    subplot(4,1,1)
160    plot(t, xt, 'k')
161    v(2) = t_max;
162    v(3) = -EfctAmp;
163    v(4) = EfctAmp;
164    axis(v);
165    title('Sinal analógico')
166    ylabel('x(t)')
167    xlabel('t (s)')
168    %
169    %
170    subplot(4,1,2)
171    plot(t, xt, 'k')
172    hold on
173    stem(n/Fs, xn, 'r')
174    %
175    v(2) = t_max;
176    v(3) = -EfctAmp;
177    v(4) = EfctAmp;
178    axis(v);
179    title('Sinal analógico e Sinal analógico amostrado')
180    ylabel('x(t) e x(nTs)')
181    xlabel('t (s)')
182    %
183    %
184    subplot(4,1,3)
185    stem(n/Fs, xn, 'r')
186    v(2) = t_max;
187    v(3) = -EfctAmp;
188    v(4) = EfctAmp;
189    axis(v);
190    title('Sinal analógico amostrado')
191    ylabel('x(nTs)')
192    xlabel('nTs (s)')
193    %
194    %

```

```

196 | subplot(4,1,4)
197 | stem(n,xn,'b')
198 | v(2) = n_max;
199 | v(3) = -EfctAmp;
200 | v(4) = EfctAmp;
201 | axis(v);
202 | title('Sinal discreto')
203 | ylabel('x[n]')
204 | xlabel('n')
205 |
206 |
207 |
208 | % EOF
209 |

```

Código 2.4: Influência do valor da taxa de amostragem.

```

1 | %%%%%%
2 | %
3 | % Arquivo: ana_smp_seq_3_Fs_snd_2021_10_30.m
4 | %
5 | %%%%%%
6 |
7 | %
8 | %
9 | %%%%%%
10 | %
11 | % Exemplo sobre:
12 | %
13 | % - Sinal senoidal analogico.
14 | % - Grafico (da aproximacao) de sinal analogico.
15 | % - Amostragem de sinal continuo.
16 | % - Sinal senoidal amostrado.
17 | % - Grafico de sinal amostrado.
18 | % - Geracao de som por interpolacao.
19 | % - Importancia da taxa de amostragem original.
20 | %
21 | % Autor : Alexandre S. de la Vega
22 | %
23 | %%%%%%
24 | %
25 | %
26 | %
27 | _____
28 | %
29 | % limpeza do ambiente de trabalho
30 | clear all
31 | close all
32 |
33 | %
34 | %
35 | _____
36 | %
37 | %
38 | %
39 | % Eh suposto um sinal senoidal analogico,
40 | % 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 .

```

```

101 %  

101 % Sao separadas as amostras x_f a partir das amostras x_rec.  

101 %  

103 %  

105 % Nf = fix( Tcos/Ts ) + 1  

105 % = fix( Fs/fcos ) + 1  

107 %  

107 Nf = fix( Fs/f ) + 1;  

109 %  

111 n_f = 0:(Nf-1);  

111 %  

113 %%x_f = A * cos( (Omega*n_f) + Theta );  

113 %  

115 x_f = x_rec(1:Nf);  

117 %  

117 % _____  

119 %  

121 %  

121 % Sao realizados diversos graficos  

123 %  

125 %  

125 % Definicoes gerais  

127 %  

129 % inicializa o numero do canvas  

129 FigNbr = 0;  

131 % calcula faixa de tempo de 1 periodo fundamental  

133 t_f = (n_f / Fs); % t = n*Ts (s)  

135 % calcula faixa de tempo de gravacao  

135 %t_rec = (n_rec / Fs); % t = n*Ts (s)  

137 %  

139 % _____  

141 %  

143 % Eh desenhado 1 periodo fundamental  

143 % (da aproximacao) do sinal analogico.  

145 %  

147 %  

147 teo_str = 'Sinal senoidal analógico: período fundamental.';  

149 %  

149 signal_str = 'x (t) = A cos ( (2 \pi f t) + \Theta )';  

151 %  

151 sep_str = ' ; ';  

153 %  

153 param_str = [ 'A = ' , num2str(A) ];  

155 param_str = [ param_str , 'f = ' , num2str(f) , ' (Hz)' ];  

155 param_str = [ param_str , 'Theta = ' , num2str(Theta) , ' (rad)' ];  

157 %

```

```

159 | ana_tit_str = {teo_str, signal_str, param_str};
160 |
161 | FigNbr = FigNbr + 1;
162 | figure(FigNbr)
163 |
164 | plot(t_f, x_f, "k")
165 |
166 | v = axis;
167 | v(2) = t_f(end);
168 | axis(v)
169 |
170 | title(ana_tit_str)
171 | ylabel('x (t)')
172 | xlabel('t (s)')
173 |
174 | % para teste...
175 | %return

176 |
177 |
178 | % -----
179 | %

180 |
181 | %
182 | % Eh desenhado 1 periodo fundamental
183 | % do sinal amostrado com Fs.
184 | %

185 |
186 | %
187 | teo_str = 'Sinal senoidal amostrado: período fundamental.';
188 | %
189 | signal_str = 'x (nTs) = A cos ((2 \pi f n Ts) + \Theta)';
190 | %
191 | sep_str = ' ; ';
192 | %
193 | param_str = [ 'A = ', num2str(A)
194 |               ];
195 | param_str = [ param_str, sep_str, 'f = ', num2str(f), '(Hz)' ];
196 | param_str = [ param_str, sep_str, '\Theta = ', num2str(Theta), '(rad)' ];
197 | param_str = [ param_str, sep_str, 'Ts = ', num2str(1/Fs), '(s)' ];
198 | param_str = [ param_str, sep_str, 'Fs = 1/Ts = ', num2str(Fs), '(Hz)' ];
199 | %
200 | smp_tit_str = {teo_str, signal_str, param_str};

201 |
202 | FigNbr = FigNbr + 1;
203 | figure(FigNbr)
204 |
205 | stem(t_f, x_f, "k")
206 |
207 | v = axis;
208 | v(2) = t_f(end);
209 | axis(v)
210 |
211 | title(smp_tit_str)
212 | ylabel('x (n Ts)')
213 | 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]')
xlabel('n')

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

```

```

273 | %      da sequencia amostrada com Fs.
273 | %
275 | %
275 | FigNbr = FigNbr + 1;
277 | figure(FigNbr)
277 | %
279 | %
279 | subplot(3,1,1)
281 | %
281 | plot(t_f, x_f, "k")
283 | %
283 | v = axis;
285 | v(2) = t_f(end);
285 | axis(v)
287 | %
287 | title({'Geração de x(n Ts) por amostragem de x(t).',
289 |         'Geração de x[n] por indexação de x(n Ts).'})
289 | %
291 | ylabel('x (t)')
291 | xlabel('t (s)')
293 | %
293 | %
295 | subplot(3,1,2)
295 | %
297 | stem(t_f, x_f, "k")
297 | %
299 | v = axis;
299 | v(2) = t_f(end);
301 | axis(v)
301 | %
303 | ylabel('x (n Ts)')
303 | xlabel('n Ts (s)')
305 | %
305 | %
307 | subplot(3,1,3)
307 | %
309 | stem(n_f, x_f, "k")
309 | %
311 | v = axis;
311 | v(2) = n_f(end);
313 | axis(v)
313 | %
315 | ylabel('x [n]')
315 | xlabel('n')
317 |
317 | % para teste...
319 | %return

321 | %
321 | % _____
323 | %

325 | %
325 | % Eh desenhado 1 periodo fundamental
327 | % conjunto:
327 | %     da recuperacao
329 | %     do sinal amostrado
329 | %     com Fs diferentes.

```

```

331 %
333 %
335 FigNbr = FigNbr + 1;
335 figure(FigNbr)
336 %
337 %
338 subplot(3,1,1)
339 %
340 stem(t_f, x_f, "k")
341 %
342 v = axis;
343 v(2) = 2*t_f(end);
344 axis(v)
345 %
346 title('Geração de x(n Ts) a partir de x[n], empregando Fs = 1/Ts.')
347 ylabel('x (n Ts)')
348 %
349 %
350 subplot(3,1,2)
351 %
352 stem((2*t_f), x_f, "k")
353 %
354 v = axis;
355 v(2) = 2*t_f(end);
356 axis(v)
357 %
358 title('Geração de x(n Ts) a partir de x[n], empregando (Fs/2) = 1/(2 Ts).')
359 ylabel('x (n Ts)')
360 %
361 %
362 subplot(3,1,3)
363 %
364 stem((0.5*t_f), x_f, "k")
365 %
366 v = axis;
367 v(2) = 2*t_f(end);
368 axis(v)
369 %
370 title('Geração de x(n Ts) a partir de x[n], empregando (2 Fs) = 1/(Ts/2).')
371 ylabel('x (n Ts)')
372 %
373 xlabel('n Ts (s)')
374 %
375 % para teste...
376 %return
377 %
378 %
379 % _____
380 %
381 %
382 %
383 % Eh desenhado 1 periodo fundamental
384 % conjunto:
385 % da recuperacao
386 % do sinal analogico
387 % com Fs diferentes.
388 %
389

```

```

%
391 FigNbr = FigNbr + 1;
figure(FigNbr)
393 %
%
395 subplot(3,1,1)
%
397 plot(t_f, x_f, "k")
%
399 v = axis;
v(2) = 2*t_f(end);
401 axis(v)
%
403 title('Geração de x(t) a partir de x[n], empregando Fs = 1/Ts.')
ylabel('x (t)')
405 %
%
407 subplot(3,1,2)
%
409 plot((2*t_f), x_f, "k")
%
411 v = axis;
v(2) = 2*t_f(end);
413 axis(v)
%
415 title('Geração de x(t) a partir de x[n], empregando (Fs/2) = 1/(2 Ts).')
ylabel('x (t)')
417 %
%
419 subplot(3,1,3)
%
421 plot((0.5*t_f), x_f, "k")
%
423 v = axis;
v(2) = 2*t_f(end);
425 axis(v)
%
427 title('Geração de x(t) a partir de x[n], empregando (2 Fs) = 1/(Ts/2).')
ylabel('x (t)')
429 %
xlabel('t (s)')
431 %
% para teste...
433 %return

435 %
%
437 %

439 %
% Eh considerada a frequencia de amostragem Fs original.
441 %
% Os valores das amostras sao enviados para a placa de som.
443 %
% A placa de som interpola os pontos com Fs e gera o som.
445 %
%
447 % Sao gerados sons com Fs diferentes da Fs original.
%

```

Código 2.5: Exemplo de amostragem de $\cos()$.

```
2 %  
3 % Arquivo: cos_amostragem.m  
4 %  
5 %%  
6 %  
7 %  
8 %%  
9 %  
10 % DSP - Aula 04/01/2016  
11 %  
12 % Autor: Alexandre Santos de la Vega  
13 %%  
14 %  
15 %
```

```

18    %%%%%% %%%%%% %%%%%% %%%%%% %%%%%%
19    %
20    % elimina as variveis de usuario , existentes no ambiente.
21    clear all
22    %
23    % elimina os graficos criados pelo usuario , existentes no ambiente.
24    close all
25    %
26    %
27    %%%%%% %%%%%% %%%%%% %%%%%% %%%%%%
28    %
29    % definicao de parametros gerais
30    %F0 = 1e3;
31    %F0 = 4.4e3;
32    %F0 = 10e3;
33    F0 = 17.6e3;
34    %
35    Fs = 44e3;
36    %
37    T0 = 1/F0;
38    Ts = 1/Fs;
39    %
40    Trec = 12*T0;
41    %
42    FigNbr = 0;
43    %
44    %
45    %%%%%% %%%%%% %%%%%% %%%%%% %%%%%%
46    %
47    %
48    % simulacao de sinal analogico
49    %
50    % definicao de parametros
51    %
52    % Obs.: NPP = V eh equivalente a Fs = V*F0 ...
53    %
54    NumPtsPer = 200;
55    Tstep = T0 / NumPtsPer;
56    nT = 0:Tstep:(Trec-Tstep);
57    %
58    % definicao da funcao
59    xa = cos(2*pi*F0*nT);
60    %
61    % grafico
62    FigNbr = FigNbr + 1;
63    figure(FigNbr)
64    %
65    plot(nT,xa,'r')
66    %
67    V = axis;
68    axis([V(1) Tstep*length(xa) V(3) V(4)])
69    %
70    title('Simulação de curva analógica')
71    ylabel('x_a (t)')
72    xlabel('t (s)')
73    %
74    %

```

```

76 |%%%%% %%%%%% %%%%%% %%%%%% %%%%%%
77 |%
78 |% simulacao de amostragem
79 |%
80 |% definicao de parametros
81 |%
82 |%
83 n = 0:fix(Trec/Ts);
84 |%
85 |% definicao da funcao
86 xd = cos(2*pi*F0*Ts*n);

87 |% grafico do sinal amostrado
88 FigNbr = FigNbr + 1;
89 figure(FigNbr)
90 |%
91 plot(nT,xa,'r')
92 hold on
93 stem(n*Ts,xd,'k')
94 |%
95 V = axis;
96 axis([V(1) Tstep*length(xa) V(3) V(4)])
97 |%
98 title('Simulacao de amostragem: sinal original e sinal amostrado')
99 ylabel('x_a (t) e x_a (n Ts)')
100 xlabel('t (s)')
101 hold off

102 |%
103 |% grafico do sinal discreto
104 FigNbr = FigNbr + 1;
105 figure(FigNbr)
106 |%
107 stem(n,xd,'k')
108 |%
109 V = axis;
110 axis([V(1) length(xd)-1 V(3) V(4)])
111 |%
112 title('Simulacao de amostragem: sinal discreto')
113 ylabel('x [n]')
114 xlabel('n')
115 |%
116 |%
117 |% reuniao de graficos
118 |%
119 |%
120 |% reuniao de graficos
121 |%
122 FigNbr = FigNbr + 1;
123 figure(FigNbr)
124 |%
125 |%
126 subplot(3,1,1)
127 plot(nT,xa,'r')
128 |%
129 V = axis;
130 axis([V(1) Tstep*length(xa) V(3) V(4)])
131 |%
132 title('Simulacao de curva analogica')
133 ylabel('x_a (t)')
134 |%

```

```

136 | xlabel( 't ( s )' )
    |
    %
138 | subplot(3,1,2)
    | plot(nT,xa,'r')
    | hold on
    | stem(n*Ts,xd,'k')
    |
142 | V = axis;
    | axis([V(1) Tstep*length(xa) V(3) V(4)])
    |
146 | title('Simulação de amostragem: sinal original e sinal amostrado')
    | ylabel('x_a ( t ) e x_a ( n Ts )')
148 | xlabel('t ( s )')
    | hold off
    |
150 | %
    |
152 | subplot(3,1,3)
    | stem(n,xd,'k')
    |
154 | V = axis;
    | axis([V(1) length(xd)-1 V(3) V(4)])
    |
158 | title('Simulação de amostragem: sinal discreto')
    | ylabel('x [ n ]')
160 | xlabel('n')
    |
162 | %
    | % EOF
164 | %

```

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.

```

2 %
% Arquivo: cadeia_proc_dig_sinal_ana.m
4 %
%
6
8 %
%
10 %
% Demos para %
12 % Apostila de DSP %
%
14 % Topico: cadeia %
% de processamento digital %
16 % de sinal analogico %
%
18 % Autor : Alexandre Santos de la Vega %
% Modifs: /2010-1/2011-2/2022-1/ %

```

```

20  %
22  %
24
26  % limpa ambiente
27  clear all
28  close all
29
30  %
31  % investiga interfaces graficas existentes
32  available_graphics_toolkits()
33  %
34  % investiga interface grafica default
35  graphics_toolkit()
36  %
37  % experimenta diferentes interfaces graficas
38  graphics_toolkit("fltk")
39  graphics_toolkit("gnuplot")
40  graphics_toolkit("qt")
41  %
42
43
44  % define parametros
45  A0 = 1;
46  F0 = 1e3;
47  T0 = 1/F0;
48  %
49  NOP = 2;      % numero de periodos visualizados
50  %
51  PPPa = 1000; % numero de pontos por periodo para sinal analogico
52  Tsa = T0/PPPa;
53  t = 0:Tsa:(NOP*T0);
54  %
55  PPPs = 10;    % numero de pontos por periodo para sinal amostrado
56  Tss = T0/PPPs;
57  nTs = 0:Tss:(NOP*T0);
58  %
59  n = 0:(length(nTs) - 1);
60
61
62  % constroi sinais
63  xa = A0*cos(2*pi*F0*t);    % sinal analogico
64  xs = A0*cos(2*pi*F0*nTs); % sinal amostrado
65
66
67  % quantiza sinais usando funcao definida externamente
68  NOI = 5; % numero de intervalos do grid
69  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
71
72
73  % desenha graficos
74  %
75  back_val = 1.0;
76
77  figure(1)
78  %

```

```

80 | subplot(3,2,1)
81 | plot(t,xa,'k')
82 | set(gca,"color",[back_val,back_val,back_val])
83 | ylabel('x(t)')
84 | xlabel('t (s)')
85 | title({'Sinal de entrada analógico', 'ou', ...
86 |       'Sinal na saída do filtro anti-aliasing'})
87 |
88 | subplot(3,2,3)
89 | plot(t,xa,'k')
90 | hold on
91 | for k = 1:(length(xs)-1)
92 | % stem(nTs(k),xs(k),':b') % não ficou bom...
93 |     plot([nTs(k) nTs(k+1)], [xs(k) xs(k)], 'b')
94 |     plot([nTs(k+1) nTs(k+1)], [xs(k) xs(k+1)], 'b')
95 | end
96 | set(gca,"color",[back_val,back_val,back_val])
97 | ylabel('r(t)')
98 | xlabel('t (s)')
99 | title('Sinal na saída do S/H')
100 |
101 | subplot(3,2,5)
102 | plot(t,xa,'k')
103 | hold on
104 | for k = 1:(length(xs)-1)
105 |     plot([nTs(k) nTs(k+1)], [xs(k) xs(k)], 'b')
106 |     plot([nTs(k+1) nTs(k+1)], [xs(k) xs(k+1)], 'b')
107 | end
108 | for k = 1:(length(xsq)-1)
109 |     plot([nTs(k) nTs(k+1)], [xsq(k) xsq(k)], 'r')
110 |     plot([nTs(k+1) nTs(k+1)], [xsq(k) xsq(k+1)], 'r')
111 | end
112 | set(gca,"color",[back_val,back_val,back_val])
113 | ylabel('c(t)')
114 | xlabel('t (s)')
115 | title('Sinal na saída do ADC')
116 |
117 | subplot(3,2,2)
118 | plot((t*F0*PPPs),xa,'k')
119 | hold on
120 | stem(n,xs,'k')
121 | stem(n,xsq,'r')
122 | set(gca,"color",[back_val,back_val,back_val])
123 | ylabel('[ x[n] ]_Q')
124 | xlabel('n')
125 | title({'Sequência digital de entrada', '=' , ...
126 |       'Sequência digital de saída'})
127 |
128 | subplot(3,2,4)
129 | stem(n,xsq,:r)
130 | hold on
131 | for k = 1:(length(xsq)-1)
132 |     plot([n(k) n(k+1)], [xsq(k) xsq(k)], 'r')
133 |     plot([n(k+1) n(k+1)], [xsq(k) xsq(k+1)], 'r')
134 | end
135 | set(gca,"color",[back_val,back_val,back_val])
136 | ylabel('c(t)')
137 | xlabel('t (s)')
138 | title('Sinal na saída do DAC')

```

```
138 %  
139 subplot(3,2,6)  
140 plot(t,xa,'k')  
141 hold on  
142 plot(nTs,xsq,'r')  
143 set(gca,"color",[back_val,back_val,back_val])  
144 ylabel('x(t)')  
145 xlabel('t (s)')  
146 title({'Sinal na saída do filtro de smoothing', '=' , ...  
147 'Sinal de saída analógico'})  
148 %  
149 % EOF  
150 %
```


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 |%%%%%%%
2 |%
3 |% Arquivo: tipos_de_seqs_sist_num.m
4 |%
5 |%%%%%%%
6 |
7 |%
8 |%%%%%%%
9 |%
10|%
11|% Demos para %
12|% Apostila de DSP %
13|%
14|% Topico: tipos de sequencias %
15|% sistema numerico %
16|%
17|% Autor : Alexandre Santos de la Vega %
18|% Modifs: /2012-02/ %
19|%
20|%%%%%%%
21|%
```

```

23
24 % limpa ambiente
25 clear all
26 close all
27
28
29 % constroi sinais
30 vr = [ 1 2 3 4 5 6 5 4 3 2 1]; % sinal real
31 wr = 4 * [ 1 1 1 0 0 0 0 -1 -1 -1]; % sinal real
32 %
33 xc = vr + j * wr; % sinal complexo
34
35 %
36 % define indice
37 n = 0:(length(vr)-1);
38
39 %
40 % desenha graficos
41 %
42 % lembrete: AXIS([XMIN XMAX YMIN YMAX]);
43 %
44 figure(1)
45 %
46 % sequencias reais
47 %
48 subplot(2,2,1)
49 stem(n,vr,'k')
50 ylabel('v_r [n]')
51 title('Sequências reais: v_r [n] e w_r [n]')
52 v = axis;
53 AXIS([v(1) v(2) -6 6]);
54 %
55 subplot(2,2,3)
56 stem(n,wr,'k')
57 ylabel('w_r [n]')
58 xlabel('n')
59 v = axis;
60 AXIS([v(1) v(2) -6 6]);
61 %
62 % sequencia complexa
63 %
64 subplot(2,2,2)
65 stem(n,abs(xc),'b')
66 ylabel('| x_c [n] |')
67 title('Sequência complexa: x_c [n] = v_r [n] + j \cdot w_r [n]')
68 v = axis;
69 AXIS([v(1) v(2) -6 6]);
70 %
71 subplot(2,2,4)
72 stem(n,angle(xc),'r')
73 ylabel('angle x_c [n] (rad)')
74 xlabel('n')
75 v = axis;
76 AXIS([v(1) v(2) -pi pi]);
77
78 %
79 % EOF
80 %
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.

```

1  %%%
2  %
3  % Arquivo: tipos_de_seqs_comprimento.m
4  %
5  %%%
6
7  %
8  %
9  %%%
10 %
11 % Demos para
12 % Apostila de DSP
13 %
14 % Topico: tipos de sequencias
15 %           comprimento
16 %
17 % Autor : Alexandre Santos de la Vega %
18 % Modifs: /2012-02/
19 %
20 %%%
21 %

23 %
24 % limpa ambiente
25 clear all
26 close all
27

29 %
30 % constroi sinais
31 %
32 xfin = [ 0 0 0 1 2 3 0 0 0 3 2 1 0 0 0 ];
33 %
34 xz = zeros(1,41);
35 %
36 xld1 = xz;
37 xld1(11:end) = 1;
38 %
39 xld2 = xz;
40 xld2(21:end) = 1;
41 %
42 xld3 = xz;
43 xld3(31:end) = 1;
44 %
45 xle1 = fliplr(xld1);
46 xle2 = fliplr(xld2);
47 xle3 = fliplr(xld3);
48 %

49 %
50 % define indices
51 %
52 nf1 = -20:-6;
53 nf2 = -7:7;
54 nf3 = 6:20;
55 
```

```

55 % ni = -20:20;
57 %
59 % desenha graficos
61 %
63 % lembrete: AXIS([XMIN XMAX YMIN YMAX]);
65 %
67 figure(1)
69 %
71 % sequencias de comprimento finito
73 %
75 subplot(3,1,1)
77 stem(nf1,xfin,'k')
79 title({'Sequências de comprimento finito — N = 15', ...
81 'x_1 [n] = x_2 [n+13] = x_3 [n+26]'})
83 ylabel('x_1 [n]')
85 v = axis;
87 axis([-20 20 v(3) v(4)]);
89 %
91 subplot(3,1,2)
93 stem(nf2,xfin,'k')
95 title('x_2 [n] = x_1 [n-13] = x_3 [n+13]')
97 ylabel('x_2 [n]')
99 xlabel('n')
101 v = axis;
103 axis([-20 20 v(3) v(4)]);
105 %
107 subplot(3,1,3)
109 stem(nf3,xfin,'k')
111 title('x_3 [n] = x_1 [n-26] = x_2 [n-13]')
113 ylabel('x_3 [n]')
115 xlabel('n')
117 v = axis;
119 axis([-20 20 v(3) v(4)]);
121 %
123 figure(2)
125 %
127 % sequencias de comprimento infinito
129 %
131 %
133 % lateral esquerda
135 %
137 subplot(3,2,1)
139 stem(ni,xle1,'k')
141 title({'Sequências de comprimento infinito', ...
143 'Lateral esquerda'})
145 ylabel('u [-n+10]')
147 %
149 subplot(3,2,3)
151 stem(ni,xle2,'k')
153 title('Lateral esquerda — Anticausal')
155 ylabel('u [-n]')
157 %
159 subplot(3,2,5)
161 stem(ni,xle3,'k')
163 title('Lateral esquerda')
165 ylabel('u [-n-10]')
167 xlabel('n')

```

```

113 %
114 % lateral direita
115 %
116 subplot(3,2,2)
117 stem(ni,xld1,'k')
118 title({'Sequências de comprimento infinito', ...
119             'Lateral direita'})
120 ylabel('u [n+10]')
121 %
122 subplot(3,2,4)
123 stem(ni,xld2,'k')
124 title('Lateral direita - Causal')
125 ylabel('u [n]')
126 %
127 subplot(3,2,6)
128 stem(ni,xld3,'k')
129 title('Lateral direita')
130 ylabel('u [n-10]')
131 xlabel('n')

132 %
133 %
134 % EOF
135 %

```

3.2.3 Simetria

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

Código 3.3: Tipos de sequências - simetria.

```

2 %
3 % Arquivo: tipos_de_seqs_simetria.m
4 %
5 %
6 %
7 %
8 %
9 %
10 %
11 % Demos para %
12 % Apostila de DSP %
13 %
14 % Topico: simetrias %
15 %
16 % Autor : Alexandre Santos de la Vega %
17 % Modifs: /2010-01/2011-02/ %
18 %
19 %
20 %

21 %
22 % limpa ambiente
23 clear all
24 close all
25
26

```

```

28 % define parametros
N = 50;
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
48 xc_cs = 0.5 * (xc + conj(fliplr(xc)));
xc_ca = 0.5 * (xc - conj(fliplr(xc)));

50
% desenha graficos
52 %
53 figure(1)
%
54 % sequencia real
55 %
56 subplot(3,1,1)
57 stem(n,xr)
58 ylabel('x[n]')
59 xlabel('n')
60 title('Sequência real: x[n] = x_e[n] + x_o[n]')
61 %
62 subplot(3,1,2)
63 stem(n,x_e)
64 ylabel('x_e[n]')
65 xlabel('n')
66 title('Sequência par: x_e[n]')
67 %
68 subplot(3,1,3)
69 stem(n,x_o)
70 ylabel('x_o[n]')
71 xlabel('n')
72 title('Sequência ímpar: x_o[n]')
73 %
74 %

76 figure(2)
%
77 % sequencia complexa
78 %
79 % modulo
80 %
81 subplot(3,2,1)
82 stem(n,real(xc))
83 ylabel('Re \{ x[n] \}')
84 xlabel('n')
85 title('Parte real da sequência complexa:')

```

```

88 %  $\operatorname{Re} \{x[n]\} = \operatorname{Re} \{x_{\text{cs}}[n] + x_{\text{ca}}[n]\}$ 
89 subplot(3,2,3)
90 stem(n, real(xc_cs))
91 ylabel('Re \{x_{\text{cs}}[n]\}')
92 xlabel('n')
93 title('Parte real da sequência conjugada simétrica:
94     Re \{x_{\text{cs}}[n]\}')
95 %
96 subplot(3,2,5)
97 stem(n, real(xc_ca))
98 ylabel('Re \{x_{\text{ca}}[n]\}')
99 xlabel('n')
100 title('Parte real da sequência conjugada anti-simétrica:
101     Re \{x_{\text{ca}}[n]\}')
102 %
103 % angulo de fase em graus
104 %
105 subplot(3,2,2)
106 stem(n, imag(xc))
107 ylabel('Im \{x[n]\}')
108 xlabel('n')
109 title('Parte imaginária da sequência complexa:
110     Im \{x[n]\} = Im \{x_{\text{cs}}[n] + x_{\text{ca}}[n]\}')
111 %
112 subplot(3,2,4)
113 stem(n, imag(xc_cs))
114 ylabel('Im \{x_{\text{cs}}[n]\}')
115 xlabel('n')
116 title('Parte imaginária da sequência conjugada simétrica:
117     Im \{x_{\text{cs}}[n]\}')
118 %
119 subplot(3,2,6)
120 stem(n, imag(xc_ca))
121 ylabel('Im \{x_{\text{ca}}[n]\}')
122 xlabel('n')
123 title('Parte imaginária da sequência conjugada anti-simétrica:
124     Im \{x_{\text{ca}}[n]\}')
125 %
126 %
127 % EOF
128 %

```

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 | %%%
2 | %
3 | % Arquivo: deslocamento_circular.m
4 | %
5 | %%%
6 |
7 | %
8 | %%%
9 | %
10| %
11| % Demos para %
12| % Apostila de DSP %
13| %
14| % Topico: deslocamento circular %
15| %
16| % Autor : Alexandre Santos de la Vega %
17| % Modifs: /2010-01/2011-02/ %
18| %
19| %%%
20| %
21|
22| %
23| % limpa ambiente
24| clear all
25| close all
26|
27| N = 10;
28| n = 0:(N-1);
29|
30| NDl = 3;
31| NDr = -3;

```

```

33 | x = n;
35 | k1 = mod((n+NDl),N);
y1 = x(k1+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,y1,'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 |
53 | subplot(2,1,2)
stem(n,x,'b')
hold on
55 | stem(n,yr,'r')
ylabel('y[n] = x[<n+N_D>_N]')
 xlabel('n')
57 | title('Deslocamento circular: direita, com N_D = 3')
legend('x[n]', 'y[n]')
59 |

61 |
62 | % EOF
63 |

```

Código 3.5: Função tenda.

```

1 |%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 |%
3 |% Arquivo: tenda.m
4 |%
5 |%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 |%
7 |function y = tenda(t)
8 |%
9 |% TENDA Gera a função exemplo tenda.
10 |%
11 |y1 = (-2*t)+(-4);
12 |y2 = (0.25*t)+(1.25);
13 |y3 = (-0.25*t)+(4.75);
14 |y4 = (-2*t)+(24);
15 |
16 |y = y1.* (2<=t & t<3) + y2.* (3<=t & t<7) + ...
17 |      y3.* (7<=t & t<11) + y4.* (11<=t & t<=12) ;
18 |%
19 |% EOF
%
```

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

```

2 %
% Arquivo: tenda_ana_desl_escal_t.m
4 %
%
6 %

8 %
%
%
10 % Titulo: Demo de deslocamentos e %
12 % escalamentos em t %
%
14 % Autor : Alexandre Santos de la Vega %
15 % Datas : /2009-02/2011-01/ %
16 %
%
18 %

20 % limpeza do ambiente de trabalho
22 % variaveis
clear all
24 % janelas
close all
26

28 % definicao dos parametros
Tmin = -5;
30 Tmax = 25;
Step = 0.1;
32 t = Tmin:Step:Tmax;
34

36 % definicao das funcoes
% nota: funcao tenda definida em arquivo tenda.m
38 Forg = tenda(t);
Fleft = tenda(t+5);
40 Fright = tenda(t-5);
Fcomp = tenda(2*t);
42 Fexpand = tenda(t/2);
Frx = tenda(2*(t-5));
44 Fxr = tenda((2*t)-5);

46 % graficos
48 %
figure(1)
50 %
subplot(3,2,1)
52 plot(t,Forg)
ylabel('f(t)')
54 title('Deslocamentos em t')
%
56 subplot(3,2,3)
plot(t,Fleft)

```

```

58 | ylabel('f_a(t) = f(t+T_D)')
59 | title('Avanço: T_D = 5 s')
60 |
61 | subplot(3,2,5)
62 | plot(t,Fright)
63 | ylabel('f_d(t) = f(t-T_D)')
64 | title('Atraso: T_D = 5 s')
65 | xlabel('t (s)')
66 |
67 | subplot(3,2,2)
68 | plot(t,Forg)
69 | ylabel('f(t)')
70 | title('Escalamentos em t')
71 |
72 | subplot(3,2,4)
73 | plot(t,Fcomp)
74 | ylabel('f_c(t) = f(t*K)')
75 | title('Compressão: K = 2')
76 |
77 | subplot(3,2,6)
78 | plot(t,Fexpand)
79 | ylabel('f_e(t) = f(t/K)')
80 | title('Expansão: K = 2')
81 | xlabel('t (s)')
82 |
83 | figure(2)
84 |
85 | subplot(3,1,1)
86 | plot(t,Forg)
87 | ylabel('f(t)')
88 | title('Deslocamentos e escalamentos em t')
89 |
90 | subplot(3,1,2)
91 | plot(t,Frx)
92 | ylabel('f_{dc}(t) = f(2*(t-T_D))')
93 | title('Atraso seguido de compressão: K = 2 e T_D = 5 s')
94 |
95 | subplot(3,1,3)
96 | plot(t,Fxr)
97 | ylabel('f_{cd}(t) = f((2*t)-T_D)')
98 | title('Compressão seguida de atraso: K = 2 e T_D = 5 s')
99 | xlabel('t (s)')
100 |
101 | %
102 | % EOF
103 | %

```

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

```

1 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 | %
3 | % Arquivo: tenda_dig_desl_escal_t.m
4 | %
5 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 |
7 | %

```

```

9 %%%%%
10 %
11 % Titulo : Demo de deslocamentos e %
12 % escalamentos em n %
13 %
14 % Autor : Alexandre Santos de la Vega %
15 % Datas : /2009-02/2011-01/
16 %
17%%%%%
18 %
19

21 % limpeza do ambiente de trabalho
22 % variaveis
23 clear all
24 % janelas
25 close all

27
28 % definicao dos parametros
29 Tmin = -5;
30 Tmax = 25;
31 Step = 0.1;
32 %
33 t = Tmin:Step:Tmax;
34 %
35 % conversao de tempo para indice
36 n = 0:(length(t)-1);
37 n = n + (Tmin/Step);

39
40 % definicao das funcoes
41 % nota: funcao tenda definida em arquivo tenda.m
42 Forg = tenda(t);
43 Fleft = tenda(t+5);
44 Fright = tenda(t-5);
45 Fcomp = tenda(2*t);
46 Fexpand = tenda(t/2); % 'isso representa up-sampling + interpolacao !!!'
47 Frx = tenda(2*(t-5));
48 Fxr = tenda((2*t)-5);
49 %
50 % retirando a interpolacao...
51 Finterp = Fexpand;
52 Fexpand = zeros(1,length(t));
53 odd_ind = 1:2:length(t);
54 Fexpand(odd_ind) = Finterp(odd_ind);
55

57 % graficos
58 %
59 figure(1)
60 %
61 subplot(3,2,1)
62 stem(n,Forg)
63 ylabel('f[n]')
64 title('Deslocamentos em n')
65 %
66 subplot(3,2,3)
67 stem(n,Fleft)

```

```

69 | ylabel('f_a[n] = f[n+N_D]')
70 | title('Avanço: N_D = 50')
71 | %
72 | subplot(3,2,5)
73 | stem(n,Fright)
74 | ylabel('f_d[n] = f[n-N_D]')
75 | title('Atraso: N_D = 50')
76 | xlabel('n')
77 | %
78 | subplot(3,2,2)
79 | stem(n,Forg)
80 | ylabel('f[n]')
81 | title('Escalamentos em n')
82 | %
83 | subplot(3,2,4)
84 | stem(n,Fcomp)
85 | ylabel('f_{ds}[n] = f[n*K]')
86 | title('Down-sampling: K = 2')
87 | %
88 | subplot(3,2,6)
89 | stem(n,Fexpand)
90 | ylabel('f_{us}[n] = f[n/K]')
91 | title('Up-sampling: K = 2')
92 | xlabel('n')
93 | %
94 | figure(2)
95 | %
96 | subplot(3,1,1)
97 | stem(n,Forg)
98 | ylabel('f[n]')
99 | title('Deslocamentos e escalamentos em n')
100 | %
101 | subplot(3,1,2)
102 | stem(n,Frx)
103 | ylabel('f_{dds}[n] = f[K*(n-N_D)]')
104 | title('Atraso seguido de down-sampling: K = 2 e N_D = 50')
105 | %
106 | subplot(3,1,3)
107 | stem(n,Fxr)
108 | ylabel('f_{dsd}[n] = f[(K*n)-N_D]')
109 | title('Down-sampling seguido de atraso: K = 2 e N_D = 50')
110 | xlabel('n')
111 | %
112 | % EOF
113 | %

```

Código 3.8: Deslocamento linear e espelhamento circular.

```

1 | %%%%%%%%%
2 | %
3 | % Arquivo: desloc_linear_espelh_circ.m
4 | %
5 | %%%%%%%%%
6 |
7 | %

```

```

9 | %%%%%%
10| %
11| % Titulo : Demo de deslocamento linear %
12| % e %
13| % espelhamento circular %
14| %
15| % Autor : Alexandre Santos de la Vega %
16| % Datas : /2012-01/
17| %
18| %%%%%%
19| %

21| %%%%%%
22| % limpeza do ambiente de trabalho
23| %%%%%%
24|
25| % variaveis
26| clear all
27| % janelas
28| close all

31| %%%%%%
32| % calculos
33| %%%%%%
34|
35| %
36|
37| x = [ 2; 1; 2; -2; ...
38|      -1; -2; 1; 2; ...
39|      1; -1; -2; -1; ...
40|      2; 0; -2; -1; ...
41|      0; 1; -2; 0; ...
42|      2; 1; 0; -1];
43|
44| %
45| N = 4;
46| Lx = length(x);
47|
48| %
49| xm = reshape(x, N, (Lx/N));
50|
51| %
52| ym = [xm; xm(1,:)];
53| ym = flipud(ym);
54| ym = ym(1:(end-1),:);
55|
56| %
57| y = reshape(ym,Lx,1);

58| %
59| nx = 0:(Lx-1);
60| nm = 0:(size(xm,1)-1);

61|
62| %%%%%%
63| %
64| % figuras
65| %%%%%%
66|
67|

```

```

69  %
70  FigInd = 0;
71  YlabelStrX = [ 'x_0[n]' ; 'x_1[n]' ; 'x_2[n]' ;
72    'x_3[n]' ; 'x_4[n]' ; 'x_5[n]' ];
73  YlabelStrY = [ 'y_0[n]' ; 'y_1[n]' ; 'y_2[n]' ;
74    'y_3[n]' ; 'y_4[n]' ; 'y_5[n]' ];

75  %
76  FigInd = FigInd + 1;
77  figure(FigInd)
78  stem(nx,x)
79  ylabel('x[n]')
80  xlabel('n')
81  %

82  FigInd = FigInd + 1;
83  figure(FigInd)
84  %
85  %
86  for SubInd = 1:6
87    subplot(2,3,SubInd)
88    stem(nm,xm(:,SubInd))
89    ylabel(YlabelStrX(SubInd,:))
90    xlabel('n')
91    AV = axis;
92    axis([AV(1) AV(2) -2 2]);
93 end

94  %
95  FigInd = FigInd + 1;
96  figure(FigInd)
97  %
98  for SubInd = 1:6
99    subplot(2,3,SubInd)
100   stem(nm,ym(:,SubInd))
101   ylabel(YlabelStrY(SubInd,:))
102   xlabel('n')
103   AV = axis;
104   axis([AV(1) AV(2) -2 2]);
105 end

106  %

107  %
108  FigInd = FigInd + 1;
109  figure(FigInd)
110  stem(nx,y)
111  ylabel('y[n]')
112  xlabel('n')

113  %

114  %

115  %

116  %
117  % EOF
118  %

```

Código 3.9: Sequência $x[n]$ finita.

```

2 |%%%%%%%%%%%%%
2 |%
2 |% Arquivo: x_seq.m

```

```

4  %
5  %
6  %
7  function s = x_seq( t )
8  %
9  %
10 % x_seq      Gera sequencia finita x[n].
11 %
12 %
13 % define o sinal
14 x0 = 1;
15 x1 = 2;
16 x2 = 2;
17 x3 = 1;
18 x4 = 0;

20 % gera sinal com indices originais
21 st = (x0*(t==0) + x1*(t==1) + x2*(t==2) + x3*(t==3) + x4*(t==4));
22 %
23 % descobre indices nao discretos de t
24 not_disc_ind = find(round(t) ~= t);

26 % indica indices que deverao ser ignorados
27 st(not_disc_ind) = NaN;
28 %
29 % retorna sinal com indices inteiros
30 s = st';

32 %
33 % EOF
34 %

```

Código 3.10: Sequência $h[n]$ finita.

```

2  %
3  %
4  % Arquivo: h_seq.m
5  %
6  %
7  function s = h_seq( t )
8  %
9  %
10 % h_seq      Gera sequencia finita h[n].
11 %

12 %
13 % define o sinal
14 h0 = 0;
15 h1 = 1;
16 h2 = 2;
17 h3 = 0;
18 h4 = 0;

20 % gera sinal com indices originais
21 st = (h0*(t==0) + h1*(t==1) + h2*(t==2) + h3*(t==3) + h4*(t==4));
22 %
23 % descobre indices nao discretos de t

```

```

24 | not_disc_ind = find(round(t) ~=~ t);
26 | % indica indices que deverao ser ignorados
| st(not_disc_ind) = NaN;
28 |
29 | % retorna sinal com indices inteiros
30 | s = st';
32 |
33 | %
34 | %

```

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

```

1 |%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 |%
3 |% Arquivo: x_seq_per.m
4 |%
5 |%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 |%
7 |function s = x_seq_per(Nf,D,NoP)
8 |
9 |%
10 |% x_seq_per      Gera sequencia periodica x[n].
11 |%
12 |%
13 |% define o sinal
14 |n = 0:(Nf-1);
15 |v = x_seq(n);
16 |if (NoP < 0)
17 |    v = flipud(v);
18 |    v = circshift(v,1);
19 |    NoP = -NoP;
20 |end
21 |vp = [v v v];
22 |vpd = circshift(vp,D);
23 |
24 |% retorna sinal com indices inteiros
25 |s = reshape(vpd, (NoP*Nf), 1);
26 |
27 |%
28 |%
29 |%

```

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

```

1 |%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 |%
3 |% Arquivo: h_seq_per.m
4 |%
5 |%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 |%
7 |function s = h_seq_per(Nf,D,NoP)
8 |
9 |%

```

```

11 % h_seq_per      Gera sequencia periodica h[n].
12 %
13 % define o sinal
14 n = 0:(Nf-1);
15 v = h_seq(n);
16 if (NoP < 0)
17     v = flipud(v);
18     v = circshift(v,1);
19     NoP = -NoP;
20 end
21 vp = [v v v];
22 vpd = circshift(vp,D);
23 %
24 % retorna sinal com indices inteiros
25 s = reshape(vpd, (NoP*Nf), 1);
26 %
27 %
28 % EOF
29 %

```

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

```

1 %%%%%%
2 %
3 % Arquivo: conv_linear_periodica_circular.m
4 %
5 %%%%%%
6 %
7 %
8 %%%%%%
9 %
10 % Titulo: Demo de relacionamento
11 % das convolucoes
12 % linear, periodica e circular
13 %%%%%%
14 % Autor : Alexandre Santos de la Vega
15 % Datas : /2012-01/
16 %%%%%%
17 %
18 %%%%%%
19 %

21 %%%%%%
22 % limpeza do ambiente de trabalho
23 %%%%%%
24 %
25 % variaveis
26 clear all
27 % janelas
28 close all
29 %

31 %%%%%%
32 % Constantes para graficos
33 %%%%%%

```

```

35 %
36 FigInd = 0;
37
38 %%%%
39 %% Constantes para convolucao linear
40 %%%%%%%
41 %%%
42 N = 7;
43 D = 7;
44
45
46 %%%%
47 %% x e h originais
48 %%%%%%%
49 %%%
50 FigInd = FigInd + 1;
51 figure(FigInd)
52
53 %
54 n = 0:(N-1);
55
56 %
57 xn = x_seq(n);
58
59 subplot(2,1,1)
60 stem(n,xn)
61 title('Entrada')
62 ylabel('x[n]')
63 xlabel('n')
64
65 %
66 hn = h_seq(n);
67
68 subplot(2,1,2)
69 stem(n,hn)
70 title('Resposta ao impulso')
71 ylabel('h[n]')
72 xlabel('n')
73
74
75 %%%%
76 %% x espelhados e deslocados
77 %%%%%%%
78 %%%
79 FigInd = FigInd + 1;
80 figure(FigInd)
81
82 %
83 YlabelStrX = [ 'x[-k]' ; 'x[-k+1]' ; 'x[-k+2]' ;
84 'x[-k+3]' ; 'x[-k+4]' ; 'x[-k+5]' ;
85 'x[-k+6]' ;
86 ];
87 k = -(D-1):(D-1);
88
89 %
90 xk = x_seq(k);
91
92 subplot(2,4,1)

```

```

95  stem(k,xk)
96  title('Entrada')
97  ylabel('x[k]')
98  xlabel('k')
99  AV = axis;
100 axis([-6 6 0 2]);

101 %
102 mxd = [];
103 for d = 0:(D-1)
104     mxd = [mxd x_seq(-k + d)];
105 end
106 %
107 for SubInd = 0:(D-1)
108     subplot(2,4,(SubInd+2))
109     stem(k,mxd(:,(SubInd+1)))
110     ylabel(YlabelStrX((SubInd+1),:))
111     xlabel('k')
112     AV = axis;
113     axis([-6 6 0 2]);
114 end
115

116 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
117 % h espelhados e deslocados
118 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
119 %

120 FigInd = FigInd + 1;
121 figure(FigInd)
122 %
123 YlabelStrH = [ 'h[-k]' ; 'h[-k+1]' ; 'h[-k+2]' ;
124             'h[-k+3]' ; 'h[-k+4]' ; 'h[-k+5]' ;
125             'h[-k+6]' ];
126 k = -(D-1):(D-1);

127 %
128 hk = h_seq(k);
129 %
130 subplot(2,4,1)
131 stem(k,hk)
132 title('Resposta ao impulso')
133 ylabel('h[k]')
134 xlabel('k')
135 AV = axis;
136 axis([-6 6 0 2]);

137 %
138 mhd = [];
139 for d = 0:(D-1)
140     mhd = [mhd h_seq(-k + d)];
141 end
142 %
143 for SubInd = 0:(D-1)
144     subplot(2,4,(SubInd+2))
145     stem(k,mhd(:,(SubInd+1)))
146     ylabel(YlabelStrH((SubInd+1),:))

```

```

153 |     xlabel( 'k' )
154 |     AV = axis;
155 |     axis([-6 6 0 2]);
156 | end
157 |
158 |
159 %%%%%%
160 % convolucao linear de x com h espelhados e deslocados
161 %%%%%%
162 %
163 FigInd = FigInd + 1;
164 figure(FigInd)
165 %
166 %
167 YlabelStrXHd = [ 'x[k] \cdot h[-k]' ; 'x[k] \cdot h[-k+1]' ;
168             '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]' ;
170             'x[k] \cdot h[-k+6]' ,
171         ];
172 k = -(D-1):(D-1);
173 %
174 myxhd = [];
175 for ind = 1:D
176     myxhd = [myxhd (xk .* mhd(:,ind))];
177 end
178 %
179 for SubInd = 0:(D-1)
180     subplot(2,4,(SubInd+1))
181     stem(k,myxhd(:,(SubInd+1)))
182     ylabel(YlabelStrXHd((SubInd+1),:))
183     xlabel('k')
184     AV = axis;
185     axis([-6 6 0 6]);
186 end
187 %
188 y = [];
189 for ind = 1:D
190     y = [y sum(myxhd(:,ind))];
191 end
192 %
193 subplot(2,4,8)
194 stem(n,y)
195 title('Convolução linear')
196 ylabel('y_L[n]')
197 xlabel('n')
198 AV = axis;
199 axis([-6 6 0 6]);
200 %
201 %
202 %
203 %
204 %
205 %%%%%%
206 % convolucao linear de h com x espelhados e deslocados
207 %%%%%%
208 %
209 FigInd = FigInd + 1;
210 figure(FigInd)
211

```

```

213 %  

214 YlabelStrHXd = [ 'h[k] \cdot x[-k]' , ; 'h[k] \cdot x[-k+1]', ;  

215 'h[k] \cdot x[-k+2]', ; 'h[k] \cdot x[-k+3]', ;  

216 'h[k] \cdot x[-k+4]', ; 'h[k] \cdot x[-k+5]', ;  

217 'h[k] \cdot x[-k+6]', ;  

218 k = -(D-1):(D-1);  

219 %  

220 myhxd = [];  

221 for ind = 1:D  

222 myhxd = [myhxd (hk .* mxd(:,ind))];  

223 end  

224 %  

225 for SubInd = 0:(D-1)  

226 subplot(2,4,(SubInd+1))  

227 stem(k,myhxd(:,(SubInd+1)))  

228 ylabel(YlabelStrHXd((SubInd+1),:))  

229 xlabel('k')  

230 AV = axis;  

231 axis([-6 6 0 6]);  

232 end  

233 %  

234 y = [];  

235 for ind = 1:D  

236 y = [y sum(myhxd(:,ind))];  

237 end  

238 %  

239 subplot(2,4,8)  

240 stem(n,y)  

241 title('Convolução linear')  

242 ylabel('y_L[n]')  

243 xlabel('n')  

244 AV = axis;  

245 axis([-6 6 0 6]);  

246 %  

247  

248 %%%%%%  

249 % Constantes para convolução periódica  

250 % com número insuficiente de amostras  

251 %%%%%%  

252 %  

253 Np = 4;  

254 %  

255 %%%%%%  

256 % x e h periódicos  

257 % Np = 4  

258 %%%%%%  

259 %  

260 FigInd = FigInd + 1;  

261 figure(FigInd)  

262 %  

263 np = -Np:(2*Np - 1);  

264 RedLine = (Np+1):(2*Np);  

265 NoP = 3;

```

```

271 %
273 xpn = x_seq_per(Np,0 ,NoP );
%
275 subplot(2 ,1 ,1)
stem(np ,xpn)
hold on
stem(np( RedLine) ,xpn( RedLine) , 'r ')
title( 'Entrada periódica')
ylabel( 'x_p[n]')
xlabel( 'n')

283 %
hpн = h_seq_per(Np,0 ,NoP );
%
285 subplot(2 ,1 ,2)
stem(np ,hpн)
hold on
stem(np( RedLine) ,hpн( RedLine) , 'r ')
title( 'Resposta ao impulso periódica')
ylabel( 'h_p[n]')
xlabel( 'n')

293

295 %%%%%%%%%%%%%%
% x periodicos espelhados e deslocados
297 % Np = 4
%%%%%%%%%%%%%
299 %
FigInd = FigInd + 1;
figure(FigInd)

303 %
YlabelStrXp = [ 'x_p[-k]' ; 'x_p[-k+1]' ; 'x_p[-k+2]' ;
305 'x_p[-k+3]' ;
];
307 k = np;

309 %
xpк = xpn;
311 %
subplot(2 ,3 ,1)
313 stem(np ,xpк)
hold on
315 stem(np( RedLine) ,xpк( RedLine) , 'r ')
title( 'Entrada periódica')
317 ylabel( 'x_p[k]')
xlabel( 'k')
319 AV = axis;
axis([-Np (2*Np - 1) 0 2]);
321 %

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

```

```

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

340
341 %%%%%%%%%%%%%%
342 % h periodicos espelhados e deslocados
343 % Np = 4
344 %%%%%%%%%%%%%%
345 %
346 FigInd = FigInd + 1;
347 figure(FigInd)

348 %
349 YlabelStrHp = [ 'h_p[-k]' ; 'h_p[-k+1]' ; 'h_p[-k+2]' ;
350           'h_p[-k+3]' ;
351 ];
352 k = np;

353 %

354 hpk = hpn;
355 %
356 subplot(2,3,1)
357 stem(np,hpk)
358 hold on
359 stem(np(RedLine),hpk(RedLine), 'r')
360 title('Resposta ao impulso periódica')
361 ylabel('h_p[k]')
362 xlabel('k')
363 AV = axis;
364 axis([-Np (2*Np - 1) 0 2]);
365
366 %
367 NoP = -3;
368 mhpd = [];
369 for d = 0:(Np-1)
370   mhpd = [mhpd h_seq_per(Np,d,NoP)];
371 end
372 %
373 for SubInd = 0:(Np-1)
374   subplot(2,3,(SubInd+2))
375   stem(k,mhpd(:,(SubInd+1)))
376   hold on
377   stem(k(RedLine),mhpd(RedLine,(SubInd+1)), 'r')
378   ylabel(YlabelStrHp((SubInd+1),:))
379   xlabel('k')
380 AV = axis;
381 axis([-Np (2*Np - 1) 0 2]);
382 end

383
384 %%%%%%%%%%%%%%
385 % convolucao periodica de x com h espelhados e deslocados

```

```

389 % Np = 4
390 % convolucao periodica de h com x espelhados e deslocados
391 %
392 FigInd = FigInd + 1;
393 figure(FigInd)

395 %
396 YlabelStrXpHpd = [ 'x_p[k] \cdot h_p[-k]' ; 'x_p[k] \cdot h_p[-k+1]' ;
397 %                                'x_p[k] \cdot h_p[-k+2]' ; 'x_p[k] \cdot h_p[-k+3]' ;
398 ] ;
399 k = 0:(Np-1);
400 kind = k+1;
401 %
402 myxphpd = [] ;
403 for ind = 1:Np
404     myxphpd = [myxphpd (xpk(kind) .* mhpd(kind ,ind ))];
405 end
406 %
407 for SubInd = 0:(Np-1)
408     subplot(2,4,(SubInd+1))
409     stem(k ,myxphpd(kind ,(SubInd+1)))
410     ylabel(YlabelStrXpHpd((SubInd+1),:))
411     xlabel('k')
412     AV = axis;
413     axis([0 (Np - 1) 0 6]);
414 end

416 %
417 y = [] ;
418 for ind = 1:Np
419     y = [y sum(myxphpd(kind ,ind ))];
420 end
421 %
422 subplot(2,4,(Np+4))
423 stem(k,y)
424 title('Convolução periódica (N_p = 4)')
425 ylabel('y_p[n]')
426 xlabel('n')
427 AV = axis;
428 axis([0 (Np - 1) 0 6]);
429 %

430 %

431 %

432 % convolucao periodica de h com x espelhados e deslocados
433 % Np = 4
434 %
435 FigInd = FigInd + 1;
436 figure(FigInd)

438 %
439 YlabelStrHpXpd = [ 'h_p[k] \cdot x_p[-k]' ; 'h_p[k] \cdot x_p[-k+1]' ;
440 %                                'h_p[k] \cdot x_p[-k+2]' ; 'h_p[k] \cdot x_p[-k+3]' ;
441 ] ;
442 k = 0:(Np-1);
443 kind = k+1;
444 %

```

```

449 %  

450 myhpYPD = [];  

451 for ind = 1:Np  

452     myhpYPD = [myhpYPD (hpk(kind) .* mxdp(kind ,ind ))];  

453 end  

454 %  

455 for SubInd = 0:(Np-1)  

456     subplot(2,4,(SubInd+1))  

457     stem(k,myhpYPD(kind ,(SubInd+1)))  

458     ylabel(YlabelStrHpXpd((SubInd+1),:))  

459     xlabel('k')  

460     AV = axis;  

461     axis([0 (Np - 1) 0 6]);  

462 end  

463 %  

464 y = [];  

465 for ind = 1:Np  

466     y = [y sum(myhpYPD(kind ,ind ))];  

467 end  

468 %  

469 subplot(2,4,(Np+4))  

470 stem(k,y)  

471 title('Convolução periódica (N_p = 4)')  

472 ylabel('y_p[n]')  

473 xlabel('n')  

474 AV = axis;  

475 axis([0 (Np - 1) 0 6]);  

476 %  

477  

478 %%%%%%  

479 % Constantes para convolucao periodica  

480 % com numero suficiente de amostras  

481 %%%%%%  

482 %  

483 Np = 6;  

484  

485 %%%%%%  

486 % x e h periodicos  

487 % Np = 6  

488 %%%%%%  

489 %  

490 FigInd = FigInd + 1;  

491 figure(FigInd)  

492  

493 %  

494 np = -Np:(2*Np - 1);  

495 RedLine = (Np+1):(2*Np);  

496 NoP = 3;  

497  

498 %  

499 xpn = x_seq_per(Np,0,NoP);  

500 %  

501 subplot(2,1,1)  

502 stem(np,xpn)  

503 hold on  

504 stem(np(RedLine),xpn(RedLine),'r')

```

```

507 | title( 'Entrada periódica' )
508 | ylabel( 'x_p[n]' )
509 | xlabel( 'n' )

511 |
512 | hpn = h_seq_per(Np,0,NoP);
513 |
514 | subplot(2,1,2)
515 | stem(np,hpn)
516 | hold on
517 | stem(np(RedLine),hp(RedLine),'r')
518 | title( 'Resposta ao impulso periódica' )
519 | ylabel( 'h_p[n]' )
520 | xlabel( 'n' )
521 |

523 %%%%
524 % x periodicos espelhados e deslocados
525 % Np = 6
526 %%%%%
527 %
528 FigInd = FigInd + 1;
529 figure(FigInd)

531 %
532 YlabelStrXp = [ 'x_p[-k]' ; 'x_p[-k+1]' ; 'x_p[-k+2]' ;
533 'x_p[-k+3]' ; 'x_p[-k+4]' ; 'x_p[-k+5]' ];
534 k = np;

537 %
538 xpk = xpn;
539 %
540 subplot(2,4,1)
541 stem(np,xpk)
542 hold on
543 stem(np(RedLine),xp(RedLine),'r')
544 title( 'Entrada periódica' )
545 ylabel( 'x_p[k]' )
546 xlabel( 'k' )
547 AV = axis;
548 axis([-Np (2*Np - 1) 0 2]);
549 %

550 NoP = -3;
551 mxpd = [];
552 for d = 0:(Np-1)
553     mxpd = [mxpd x_seq_per(Np,d,NoP)];
554 end
555 %

556 for SubInd = 0:(Np-1)
557     subplot(2,4,(SubInd+2))
558     stem(k,mxpd(:,(SubInd+1)))
559     hold on
560     stem(k(RedLine),mxpd(RedLine,(SubInd+1)),'r')
561     ylabel(YlabelStrXp((SubInd+1),:))
562     xlabel( 'k' )
563     AV = axis;
564     axis([-Np (2*Np - 1) 0 2]);

```

```

567    end
568
569    %%%%%%%%%%%%%%
570    % h periodicos espelhados e deslocados
571    % Np = 6
572    %%%%%%%%%%%%%%
573    %
574    FigInd = FigInd + 1;
575    figure(FigInd)

576    %
577    YlabelStrHp = [ 'h_p[-k]' ; 'h_p[-k+1]' ; 'h_p[-k+2]' ;
578                  'h_p[-k+3]' ; 'h_p[-k+4]' ; 'h_p[-k+5]' ];
579    k = np;

580    %
581    hpk = hpn;
582    %
583    subplot(2,4,1)
584    stem(np,hpk)
585    hold on
586    stem(np(RedLine),hpk(RedLine),'r')
587    title('Resposta ao impulso periódica')
588    ylabel('h_p[k]')
589    xlabel('k')
590    AV = axis;
591    axis([-Np (2*Np - 1) 0 2]);
592
593    %
594    NoP = -3;
595    mhpd = [];
596    for d = 0:(Np-1)
597        mhpd = [mhpd h_seq_per(Np,d,NoP)];
598    end
599    %
600    for SubInd = 0:(Np-1)
601        subplot(2,4,(SubInd+2))
602        stem(k,mhpd(:,(SubInd+1)))
603        hold on
604        stem(k(RedLine),mhpd(RedLine,(SubInd+1)),'r')
605        ylabel(YlabelStrHp((SubInd+1),:))
606        xlabel('k')
607        AV = axis;
608        axis([-Np (2*Np - 1) 0 2]);
609    end
610
611    %
612    FigInd = FigInd + 1;
613    figure(FigInd)

614    %
615    % convolucao periodica de x com h espelhados e deslocados
616    % Np = 6
617    %%%%%%%%%%%%%%
618    %
619    FigInd = FigInd + 1;
620    figure(FigInd)

621    %
622    YlabelStrXpHpd = [ 'x_p[k] \cdot h_p[-k]' ; 'x_p[k] \cdot h_p[-k+1]' ;
623

```

```

625      'x_p[k] \cdot h_p[-k+2]', ; 'x_p[k] \cdot h_p[-k+3]', ;
627      'x_p[k] \cdot h_p[-k+4]', ; 'x_p[k] \cdot h_p[-k+5]', ;
];
628 k = 0:(Np-1);
629 kind = k+1;
630 %
631 myxphpd = [];
632 for ind = 1:Np
633     myxphpd = [myxphpd (xpk(kind) .* mhpdkind, ind))];
634 end
635 %
636 for SubInd = 0:(Np-1)
637     subplot(2,4,(SubInd+1))
638     stem(k,myxphpd(kind,(SubInd+1)))
639     ylabel(YlabelStrXpHpd((SubInd+1),:))
640     xlabel('k')
641     AV = axis;
642     axis([0 (Np - 1) 0 6]);
643 end
644 %
645 y = [];
646 for ind = 1:Np
647     y = [y sum(myxphpd(kind, ind))];
648 end
649 %
650 subplot(2,4,(Np+2))
651 stem(k,y)
652 title('Convolução periódica (N_p = 6)')
653 ylabel('y_p[n]')
654 xlabel('n')
655 AV = axis;
656 axis([0 (Np - 1) 0 6]);
657 %
658 %

659 %

660 %%%%%%
661 %%%%%%
662 % convolucao periodica de h com x espelhados e deslocados
663 % Np = 6
664 %%%%%%
665 %%%%%%
666 %

667 FigInd = FigInd + 1;
668 figure(FigInd)
669 %
670 YlabelStrHpXpd = [ 'h_p[k] \cdot x_p[-k ]', ; 'h_p[k] \cdot x_p[-k+1]', ;
671             'h_p[k] \cdot x_p[-k+2]', ; 'h_p[k] \cdot x_p[-k+3]', ;
672             'h_p[k] \cdot x_p[-k+4]', ; 'h_p[k] \cdot x_p[-k+5]', ;
];
673 k = 0:(Np-1);
674 kind = k+1;
675 %
676 myhpxpd = [];
677 for ind = 1:Np
678     myhpxpd = [myhpxpd (hpk(kind) .* mxpd(kind, ind))];
679 end
680 %
681
682 %

```

```

685 | for SubInd = 0:(Np-1)
686 |   subplot(2,4,(SubInd+1))
687 |   stem(k,myhpYPD(kind,(SubInd+1)))
688 |   ylabel(YlabelStrHPXPD((SubInd+1),:))
689 |   xlabel('k')
690 |   AV = axis;
691 |   axis([0 (Np - 1) 0 6]);
692 | end

693 |
694 | %
695 | y = [];
696 | for ind = 1:Np
697 |   y = [y sum(myhpYPD(kind,ind))];
698 | end
699 |
700 | subplot(2,4,(Np+2))
701 | stem(k,y)
702 | title('Convolução periódica (N_p = 6)')
703 | ylabel('y_p[n]')
704 | xlabel('n')
705 | AV = axis;
706 | axis([0 (Np - 1) 0 6]);
707 | %

708 | %%%%%%
709 | % Constantes para convolucao periodica
710 | % com numero insuficiente de amostras
711 | %%%%%%
712 | %
713 | Nc = 4; % Nesse exercicio, usar 2 < N < 8.
714 |

715 | %%%%%%
716 | % convolucao circular de x com h circulante
717 | %%%%%%
718 | %
719 | Nc = 4
720 | %%%%%%
721 | %
722 | FigInd = FigInd + 1;
723 | figure(FigInd)

724 | %
725 | ind = 1:Nc;
726 | n = (ind-1);

727 | %
728 | xc = xn(ind);
729 | %
730 | hc = hn(ind);
731 | c = hc;
732 | for i = 2:Nc
733 |   hc = circshift(hc,[1 0]);
734 |   c = [c hc];
735 | end
736 | %
737 | yc = c*xc;
738 | %
739 | stem(n,yc)
740 | title('Convolução circular com h[n] circulante (N_c = 4)')

```

```

743 | ylabel('y_c[n]')
744 | xlabel('n')
745 | AV = axis;
746 | axis([0 5 0 6]);
747 |
748 |
749 | %%%%%%
750 | % convolucao circular de h com x circulante
751 | % Nc = 4
752 | %%%%%%
753 | %
754 | FigInd = FigInd + 1;
755 | figure(FigInd)
756 |
757 | %
758 | ind = 1:Nc;
759 | n = (ind-1);
760 |
761 | %
762 | hc = hn(ind);
763 | %
764 | xc = xn(ind);
765 | c = xc;
766 | for i = 2:Nc
767 |     xc = circshift(xc,[1 0]);
768 |     c = [c xc];
769 | end
770 | %
771 | yc = c*hc;
772 | %
773 | stem(n,yc)
774 | title('Convolução circular com x[n] circulante (N_c = 4)')
775 | ylabel('y_c[n]')
776 | xlabel('n')
777 | AV = axis;
778 | axis([0 5 0 6]);
779 |
780 |
781 | %%%%%%
782 | % Constantes para convolucao periodica
783 | % com numero suficiente de amostras
784 | %%%%%%
785 | %
786 | Nc = 6; % Nesse exercicio , usar 2 < N < 8.
787 |
788 |
789 | %%%%%%
790 | % convolucao circular de x com h circulante
791 | % Nc = 6
792 | %%%%%%
793 | %
794 | FigInd = FigInd + 1;
795 | figure(FigInd)
796 |
797 | %
798 | ind = 1:Nc;
799 | n = (ind-1);
800 | %

```

```

803 xc = xn(ind);
804 %
805 hc = hn(ind);
806 c = hc;
807 for i = 2:Nc
808     hc = circshift(hc,[1 0]);
809     c = [c hc];
810 end
811 %
812 yc = c*xc;
813 stem(n,yc)
814 title('Convolução circular com h[n] circulante (N_c = 6)')
815 ylabel('y_c[n]')
816 xlabel('n')
817 AV = axis;
818 axis([0 5 0 6]);
819 %

820%%%%%%%%%%%%%
821 % convolucao circular de h com x circulante
822 % Nc = 6
823 %%%%%%
824 %
825 FigInd = FigInd + 1;
826 figure(FigInd)

827 %
828 ind = 1:Nc;
829 n = (ind-1);

830 %
831 hc = hn(ind);
832 %
833 xc = xn(ind);
834 c = xc;
835 for i = 2:Nc
836     xc = circshift(xc,[1 0]);
837     c = [c xc];
838 end
839 %
840 yc = c*hc;
841 %
842 stem(n,yc)
843 title('Convolução circular com x[n] circulante (N_c = 6)')
844 ylabel('y_c[n]')
845 xlabel('n')
846 AV = axis;
847 axis([0 5 0 6]);
848 %

849 %
850 % EOF
851 %

```

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.

```

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %
4 % Arquivo: seq_bas_menu.m
5 %
6 %
7 %
8 % clear all
9 % close all
10 %
11 %
12 menu_choice = 1;
13 while (menu_choice ~= 0)
14     for i = 1:50
15         disp(' ')
16     end
17     disp('* **** *')
18     disp('* Geração de sequências básicas *')
19     disp('* **** *')
20     disp(' ')
21     disp(' Opções: ')
22     disp('      00 - Sair ')
23     disp('      01 - Degrau unitário ')
24     disp('      02 - Dirichlet unitária ')
25     disp('      03 - Gate retangular unitário ')
26     disp('      04 - Impulso unitário ')
27     disp('      05 - Sequência linear unitária ')
28     disp('      06 - Módulo unitário ')
29     disp('      07 - Exponencial unitária ')
30     disp('      08 - Rampa unitária ')

```

```

32 |     disp( ' 09 - Signum unitário ')
33 |     disp( ' 10 - Sinc unitária ')
34 |     disp( ' ')
35 |     menu_choice = input( ' Escolha: ');
36 |
37 |     switch menu_choice
38 |         case {0}
39 |             disp( ' ')
40 |             disp( ' Bye... ')
41 |             disp( ' ')
42 |         case {1}
43 |             disp( ' Degrau unitário ')
44 |             t = input( ' Faixa de tempo no formato INÍCIO:FIM =
45 |             ');
46 |             stem(t ,degrau_unitario(t ))
47 |             title('Degrau unitário ')
48 |             ylabel('u [n]')
49 |             xlabel('n')
50 |         case {2}
51 |             disp( ' Dirichlet unitária ')
52 |             N = input( ' Período = ');
53 |             t = input( ' Faixa de tempo no formato INÍCIO:FIM =
54 |             ');
55 |             [s ,n] = drcl_unitaria(t ,N);
56 |             stem(t ,s)
57 |             title('Dirichlet unitária ')
58 |             ylabel('Drcl [n]')
59 |             xlabel('n')
60 |         case {3}
61 |             disp( ' Gate retangular unitário ')
62 |             Ng = input( ' Número de amostras do gate = ');
63 |             t = input( ' Faixa de tempo no formato inicio:fim
64 |             ');
65 |             stem(t ,gate_retang_unitario(Ng,t ))
66 |             title('Gate retangular unitário ')
67 |             ylabel('G_{Ng} [n]')
68 |             xlabel('n')
69 |         case {4}
70 |             disp( ' Impulso unitário ')
71 |             t = input( ' Faixa de tempo no formato INÍCIO:FIM =
72 |             ');
73 |             stem(t ,impulso_unitario(t ))
74 |             title('Impulso unitário ')
75 |             ylabel('\delta [n]')
76 |             xlabel('n')
77 |         case {5}
78 |             disp( ' Sequência linear unitária ')
79 |             t = input( ' Faixa de tempo no formato INÍCIO:FIM =
80 |             ');
81 |             stem(t ,linear_unitaria(t ))
82 |             title('Sequência linear unitária ')
83 |             ylabel('Lin [n]')
84 |             xlabel('n')
85 |         case {6}
86 |             disp( ' Módulo unitário ')
87 |             t = input( ' Faixa de tempo no formato INÍCIO:FIM =
88 |             ');

```

```

84         stem(t , modulo_unitario(t ))
85         title( 'Módulo unitário' )
86         ylabel( 'Mod [n]' )
87         xlabel( 'n' )
88     case {7}
89         disp( ' Exponencial unitária' )
90         base = input( ' Base = ' );
91         t      = input( ' Faixa de tempo no formato inicio:fim
92         =    ' ); |
93         stem(t , power_unitaria(base , t ))
94         title( 'Exponencial unitária' )
95         ylabel( 'Base^n' )
96         xlabel( 'n' )
97     case {8}
98         disp( ' Rampa unitária' )
99         t = input( ' Faixa de tempo no formato INÍCIO:FIM =
100        , ); |
101        stem(t , rampa_unitaria(t ))
102        title( 'Rampa unitária' )
103        ylabel( 'Rmp [n]' )
104        xlabel( 'n' )
105    case {9}
106        disp( ' Signum unitário' )
107        t = input( ' Faixa de tempo no formato INÍCIO:FIM =
108        , ); |
109        stem(t , signum_unitario(t ))
110        title( 'Signum unitário' )
111        ylabel( 'Sgn [n]' )
112        xlabel( 'n' )
113    case {10}
114        disp( ' Sinc unitária' )
115        t = input( ' Faixa de tempo no formato INÍCIO:FIM =
116        , ); |
117        [s,n] = sinc_unitaria(t );
118        stem(t , s)
119        title( 'Sinc unitária' )
120        ylabel( 'Sinc [n]' )
121        xlabel( 'n' )
122    otherwise
123        'do nothing... '
124    end % switch
125 end % while

126 %
127 % EOF
128 %

```

Código 3.15: Geração da função degrau unitário discreto.

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %
3 % Arquivo: degrau_unitario.m
4 %
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 %
7 function s = degrau_unitario(t)

```

```

9  %
10 % degrau_unitario      Gera função Degrau Unitario Digital.
11 %
12 %
13 % gera sinal com indices originais
14 st = 1.*(t>=0);
15 %
16 % descobre indices nao discretos de t
17 not_disc_ind = find(round(t) ~= t);
18 %
19 % indica indices que deverao ser ignorados
20 st(not_disc_ind) = NaN;
21 %
22 % retorna sinal com indices inteiros
23 s = st;
24 %
25 %
26 % EOF
27 %

```

Código 3.16: Geração da função Dirichlet unitária.

```

1  %%%%%%
2  %
3  % Arquivo: drcl_unitaria.m
4  %
5  %%%%%%
6  %
7  function [s,n] = drcl_unitaria(t,N)
8  %
9  %
10 % drcl_unitaria      Gera função Dirichlet Unitária .
11 %
12 %     É esperado: t = -Tmax ... 0 ... Tmax.
13 %
14 %
15 % descobre indice t=0
16 ind_t_0 = find(t == 0);
17 %
18 % gera nova faixa de indices
19 n = (1:length(t)) - ind_t_0;
20 %
21 % gera sinal com indices originais
22 s = diric(2*pi*t,N);
23 %
24 %
25 % EOF
26 %

```

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

```

2  %%%%%%
3  %
4  % Arquivo: gate_retang_unitario.m

```

```

4  %
5  %
6  %
7  function s = gate_retang_unitario(Ng, t)
8  %
9  %
10 % gate_retang_unitario      Gera funcao Gate Retangular Unitario Digital.
11 %
12 %
13 % gera sinal com indices originais
14 st = 1.*(- abs(Ng)<=t & t<= abs(Ng));
15 %
16 % descobre indices nao discretos de t
17 not_disc_ind = find(round(t) ~= t);
18 %
19 % indica indices que deverao ser ignorados
20 st(not_disc_ind) = NaN;
21 %
22 % retorna sinal com indices inteiros
23 s = st;
24 %
25 %
26 % EOF
27 %

```

Código 3.18: Geração da função impulso unitário discreto.

```

1  %
2  %
3  % Arquivo: impulso_unitario.m
4  %
5  %
6  %
7  function s = impulso_unitario(t)
8  %
9  %
10 % impulso_unitario      Gera funcao Impulso Unitario Digital.
11 %
12 %
13 % gera sinal com indices originais
14 st = 1.*(t==0);
15 %
16 % descobre indices nao discretos de t
17 not_disc_ind = find(round(t) ~= t);
18 %
19 % indica indices que deverao ser ignorados
20 st(not_disc_ind) = NaN;
21 %
22 % retorna sinal com indices inteiros
23 s = st;
24 %
25 %
26 % EOF
27 %

```

Código 3.19: Geração da função linear unitária discreta.

```

1 %%%%%%
2 %
3 % Arquivo: linear_unitaria.m
4 %
5 %%%%%%
6 %
7 function s = linear_unitaria(t)
8 %
9 % linear_unitaria      Gera função Linear Unitaria Digital .
10 %
11 %
12 % gera sinal com indices originais
13 st = t;
14 %
15 % descobre indices nao discretos de t
16 not_disc_ind = find(round(t) ~= t);
17 %
18 % indica indices que deverao ser ignorados
19 st(not_disc_ind) = NaN;
20 %
21 % retorna sinal com indices inteiros
22 s = st;
23 %
24 %
25 % EOF
26 %
27 %

```

Código 3.20: Geração da função módulo unitário discreto.

```

1 %%%%%%
2 %
3 % Arquivo: modulo_unitario.m
4 %
5 %%%%%%
6 %
7 function s = modulo_unitario(t)
8 %
9 % modulo_unitario      Gera função Modulo Unitario Digital .
10 %
11 %
12 % gera sinal com indices originais
13 st = -t.*(t<0) + t.*(0<=t);
14 %
15 % descobre indices nao discretos de t
16 not_disc_ind = find(round(t) ~= t);
17 %
18 % indica indices que deverao ser ignorados
19 st(not_disc_ind) = NaN;
20 %
21 % retorna sinal com indices inteiros
22 s = st;
23 %
24 %
25 % EOF
26 %
27 %

```

27 | %

Código 3.21: Geração da função potenciação unitária discreta.

```

1 %%%%%
2 %
3 % Arquivo: power_unitaria.m
4 %
5 %%%%%
6 %
7 function s = power_unitaria(base, t)
8 %
9 % power_unitaria    Gera funcao Potenciacao Unitaria Digital .
10 %
11 %
12 % gera sinal com indices originais
13 st = power(base, t);
14 %
15 % descobre indices nao discretos de t
16 not_disc_ind = find(round(t) ~= t);
17 %
18 % indica indices que deverao ser ignorados
19 st(not_disc_ind) = NaN;
20 %
21 % retorna sinal com indices inteiros
22 s = st;
23 %
24 %
25 % EOF
26 %
27 %

```

Código 3.22: Geração da função rampa unitária discreta.

```

1 %%%%%
2 %
3 % Arquivo: rampa_unitaria.m
4 %
5 %%%%%
6 %
7 function s = rampa_unitaria(t)
8 %
9 % rampa_unitaria    Gera funcao Rampa Unitaria Digital .
10 %
11 %
12 % gera sinal com indices originais
13 st = t.*(t >= 0);
14 %
15 % descobre indices nao discretos de t
16 not_disc_ind = find(round(t) ~= t);
17 %
18 % indica indices que deverao ser ignorados
19 st(not_disc_ind) = NaN;
20 %
21 %

```

```

23 % retorna sinal com indices inteiros
23 s = st;

25 %
25 % EOF
27 %

```

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

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1 %

3 % Arquivo: signum_unitario.m
3 %

5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 %

7 function s = signum_unitario(t)

9 %
9 % signum_unitario Gera funcao Signum Unitario Digital .
11 %

13 % gera sinal com indices originais
13 st = -1.*(t<0) + 1.*(0<t);

15 % descobre indices nao discretos de t
17 not_disc_ind = find(round(t) ~= t);

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

23 % retorna sinal com indices inteiros
23 s = st;

25 %
25 % EOF
27 %

```

Código 3.24: Geração da função *Sinc* unitária.

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1 %

3 % Arquivo: sinc_unitaria.m
3 %

5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 %

7 function [s,n] = sinc_unitaria(t)

9 %
9 % sinc_unitaria Gera funcao Sinc Unitaria .
11 %

13 % descobre indice t=0
13 ind_t_0 = find(t == 0);

15 % gera nova faixa de indices

```

```

17 | n = (1:length(t)) - ind_t_0;
19 | % gera sinal com indices originais
20 | s = sinc(t);
21 |
22 | %
23 | % EOF
24 | %

```

Código 3.25: Demonstração de sequência exponencial real.

```

2   %
3   % Arquivo: power_real_demo.m
4   %
5   %
6   %
7
8   %
9   %
10  %
11  %
12  % Titulo: Demo de sequencia exponencial real
13  %
14  % Autor : Alexandre Santos de la Vega
15  % Data   : 30/03/2k9 ; 08/09/2k9
16  %
17  %
18  %

20  % limpeza do ambiente de trabalho
21  % variaveis
22  clear all
23  % janelas
24  close all

26
27  % definicao dos parametros
28  n = 0:10;
29  A = 1;
30  alfa_pos_1_inf = 2;
31  alfa_pos_1 = 1;
32  alfa_pos_0_1 = 0.2;
33  alfa_neg_1_0 = - 0.2;
34  alfa_neg_1 = - 1;
35  alfa_neg_inf_1 = - 2;
36

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

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

```

```

48 | xlabel( 'n' )
48 |
49 | figure(3)
50 | stem(n, (A*power_unitaria(alfa_pos_0_1,n)))
50 | title('exponencial real: \alpha^n , \alpha = (0 ; 1)')
52 | xlabel( 'n' )

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

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

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

68 |

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

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

80 | title('exponencial real: \alpha^n')
82 |

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

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

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

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

104 | %
104 | % 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.

```
2 % Arquivo original: fasor_discreto_2021_12_06.m
3 %
4 % Arquivo: exp_visao_fasorial.m
5 %
6 %
7 %
8 %
9 %
10 %
11 %
12 % Titulo: Demo de
13 %         sequencia exponencial
14 %         interpretada como fasor discreto.
15 %
16 %         - Exp() como fasor.
17 %         - Relacao de Euler:
18 %             exp() = cos() + j sin().
19 %         - Simulacao passo a passo.
```

```

20 %           – Visões 2D e 3D para exp().
21 %           – Taxa de variação X Nf e Omega.
22 %
23 %
24 % Autor : Alexandre Santos de la Vega
25 % Data  : / 2021-01 /
26 %
27 %-----%
28 %
29 %
30 %
31 %-----%
32 %
33 %
34 %
35 clear all
36 close all
37 %
38 %-----%
39 %
40 %
41 %
42 %
43 % Definições gerais
44 %
45 %
46 %
47 mv_opt = questdlg ("Temporizado ou Manual?", ...
48 %                     "Controle do movimento de fasor discreto", ...
49 %                     "Temporizado", "Manual (teclado ou mouse)", "Cancela", ...
50 %                     "Temporizado");
51 %
52 if(strcmp (mv_opt, "Cancela"))
53 %disp ("You cancelled .");
54 return;
55 endif
56 %
57 if(strcmp (mv_opt, "Temporizado"))
58 %
59 options_cell = {"0.0", "0.5", "1.0", "1.5", "2.0", "2.5", "3.0"};
60 [sel_ind, ok] = listdlg (
61 %                         "Name", "Intervalo de tempo (s)",
62 %                         "CancelString", "Cancela",
63 %                         "ListSize", [180,130],
64 %                         "ListString", options_cell,
65 %                         "SelectionMode", "Single"
66 );
67 %
68 if (ok == 1)
69 t_pause = str2num(options_cell{sel_ind});
70 else
71 %disp ("You cancelled .");
72 return;
73 endif
74 %
75 endif
76 %
77 options_cell = {"32", "16", "8", "4", "3"};

```

```

80 [ sel_ind , ok ] = listdlg (
81   "Name" , "Período fundamental (Nf)" ,
82   "CancelString" , "Cancela" ,
83   "ListSize" , [220,100] ,
84   "ListString" , options_cell ,
85   "SelectionMode" , "Single"
86 );
87 %
88 if (ok == 1)
89   Nf = str2num(options_cell{sel_ind});
90 else
91   %disp ("You cancelled .");
92   return;
93 endif
94 %
95 Omega_Nf = (2*pi/Nf);

96 %
97 max_Nf = 0;
98 for ind=1:length(options_cell)
99   Nf_ind = str2num(options_cell{ind});
100  if (Nf_ind > max_Nf)
101    max_Nf = Nf_ind;
102  endif
103 endfor
104 %
105 %
106 % _____
107 %

108 %
109 %
110 % Construcao de
111 % Lugar Geometrico com raio constante
112 % ou
113 % Circulo de raio r
114 % ou
115 % |z| = r
116 % ou
117 % z = e^(j Omega) ; -oo < Omega < oo
118 %

119 %
120 N = 360;
121 Omega_step = 2*pi/N;
122 Omega = 0:Omega_step:(2*pi);
123 %
124 %
125 circ_1 = e^(j*Omega);
126 re_circ_1 = real(circ_1);
127 im_circ_1 = imag(circ_1);

128 %
129 %
130 % _____
131 %

132 %
133 %
134 FigNbr = 0;
135 %
136 r_lim = 1.3;

```

```

138 | c_lim = ((2*max_Nf)-1);
139 | %
140 | black_val = 0;
141 |
142 | %
143 | % -----
144 | %
145 |
146 | %
147 | FigNbr = FigNbr + 1;
148 | figure(FigNbr)
149 | %
150 | for n = 0:(Nf-1)
151 | %
152 | f_r1 = e.^ (j*Omega_Nf*n);
153 | re_f_r1 = real(f_r1);
154 | im_f_r1 = imag(f_r1);
155 |
156 | %
157 | subplot(2,2,1)
158 | %
159 | hold off
160 | %
161 | plot(re_circ_1,im_circ_1,"k")
162 | %
163 | hold on
164 | %
165 | plot(re_f_r1, im_f_r1, "ob", "markersize", 7)
166 | plot([0,re_f_r1], [0,im_f_r1], "b")
167 | %
168 | plot(re_f_r1, 0, "or", "markersize", 4)
169 | plot([0,re_f_r1], [0,0], "r")
170 | %
171 | plot(0, im_f_r1, "og", "markersize", 4)
172 | plot([0, 0], [im_f_r1,0], "g")
173 | %
174 | if (re_f_r1 > 0)
175 |   x_txt = 1.1*re_f_r1;
176 |   y_txt = 1.1*im_f_r1;
177 | elseif (abs(im_f_r1) < eps)
178 |   x_txt = 0.9*re_f_r1;
179 |   y_txt = 0.1;
180 | else
181 |   x_txt = 0.9*re_f_r1;
182 |   y_txt = 1.1*im_f_r1;
183 | end
184 | text(x_txt, y_txt, [ 'n = ', num2str(n)]);
185 | %
186 | axis([-r_lim, r_lim, -r_lim, r_lim])
187 | % axis("equal") = set(gca, 'DataAspectRatio',[1, 1, 1])
188 | axis("equal")
189 | %
190 | grid on
191 | %
192 | tit_str_1 = 'Fasor discreto: z[n] = e^{j(\Omega_0 n)}';
193 | tit_str_2 = [ '\Omega_0 = 2\pi/N_f = 2\pi / ', ...
194 |               num2str(Nf), ...
195 |               ' = ', num2str(Omega_Nf), ' rad' ];
196 | title ({tit_str_1, tit_str_2});

```

```

198  ylabel( 'y[n] = Im \{z[n]\} = sin(\Omega_0 n)')
199  xlabel( 'x[n] = Re \{z[n]\} = cos(\Omega_0 n)')
200  %
201  %
202  h_sin = subplot(2,2,2);
203  %
204  hold on
205  %
206  stem(n, im_f_r1, "og")
207  %
208  % axis([0, (Nf-1), -r_lim, r_lim])
209  axis([0, c_lim, -r_lim, r_lim])
210  axis("square")
211  %
212  grid on
213  %
214  title ('y[n] = sin(\Omega_0 n)');
215  ylabel('y[n]')
216  xlabel('n')
217  %
218  %
219  h_cos = subplot(2,2,3);
220  %
221  hold on
222  %
223  stem(n, re_f_r1, "or")
224  %
225  % axis([0, (Nf-1), -r_lim, r_lim])
226  axis([0, c_lim, -r_lim, r_lim])
227  axis("square")
228  %
229  grid on
230  %
231  %
232  title ('x[n] = cos(\Omega_0 n)');
233  ylabel('x[n]')
234  xlabel('n')
235  %
236  %view(AZIMUTH, ELEVATION)
237  view(90,90)
238  %

239  %
240  subplot(2,2,4)
241  %
242  hold on
243  %
244  %
245  % stem3(x,y,z)
246  stem3(re_f_r1, im_f_r1, n, "b")
247  %
248  axis([-r_lim, r_lim, -r_lim, r_lim, 0, (Nf-1)])
249  axis("square")
250  %
251  grid on
252  %
253  title('Curva discreta 3D: z[n] = e^{j (\Omega_0 n)}')
254  xlabel('n')
255  ylabel('y[n]')

```

```

256 xlabel('x[n]')
%
258 %view(AZIMUTH, ELEVATION)
259 view(26,18)
%
260 % fundo preto
261 %set(gca, 'color',[black_val, black_val, black_val])
%
264 %
265 if(strcmp(mv_opt, "Temporizado"))
266     pause(t_pause);
267 else
268     b = waitforbuttonpress();
269 endif
%
270 end
%
274 %
275 %msgbox("Fim do período fundamental.", "Movimento do fasor", "warn");
276 cont_opt = questdlg("Fim do período fundamental.", "Movimento do fasor", ...
277                     "Continua", "Cancela", "Continua");
%
278 if(strcmp(cont_opt, "Cancela"))
279     %disp("You cancelled.");
280     return;
281 endif
%
284 %
285 % _____
286 %
287 %
288 %
289 % completa as projeções sin() e cos()
290 % apos o periodo fundamental
291 %
292 %
293 %
294 n = 0:c_lim;
295 %
296 f_r1 = e.^ (j*Omega_Nf*n);
297 %
298 re_f_r1 = real(f_r1);
299 im_f_r1 = imag(f_r1);
300 %
301 %
302 stem(h_sin, n((Nf+1):end), im_f_r1((Nf+1):end), "ok")
303 stem(h_cos, n((Nf+1):end), re_f_r1((Nf+1):end), "ok")
304 %
305 %
306 % EOF
307 %
308

```

Código 4.2: Associação entre exponencial e fasor.

```

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %
4 % Arquivo : associa_exp_fasor.m
5 %
6 %
7 %
8 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9 %
10 %
11 % Titulo : Demo de
12 %          sequencia exponencial
13 %          interpretada como fasor
14 %          (visao 3D)
15 %
16 % Autor : Alexandre Santos de la Vega
17 % Data : / 2010-02 / 2011-01 /
18 %
19 %
20 %
21 %
22 % limpeza do ambiente de trabalho
23 % variaveis
24 clear all
25 % janelas
26 close all
27 %
28 %
29 %
30 % definicao dos parametros
NOD = 7; Theta0 = 0;
31 %NOD = 7; Theta0 = (2*pi)/((NOD+1)*2);
%NOD = 15; Theta0 = 0;
33 %NOD = 15; Theta0 = (2*pi)/((NOD+1)*2);
%
35 Omega0 = (2*pi)/(NOD+1);

37 %
38 % graficos
39 %
40 figure(1)
41 %
42 % cores
43 c_eixo = 'k';
c_disc = 'r';
45 c_cont = 'b';
%
47 % eixo
x = [0 0];
48 y = [0 0];
z = [0 NOD];
50 plot3(z,x,y,c_eixo)
hold on
51 %
52 % fasor discreto
53 x = [0 0];
y = [0 0];
54
55
56

```

```

58 | for n=0:NOD
59 |   x(2) = cos(Omega0*n + Theta0);
60 |   y(2) = sin(Omega0*n + Theta0);
61 |   z = [n n];
62 |   plot3(z,x,y,c_disc)
63 | end
64 |
65 | % fasor continuo
66 | n = (0:0.1:NOD);
67 | x = cos(Omega0*n + Theta0);
68 | y = sin(Omega0*n + Theta0);
69 | z = n;
70 | plot3(z,x,y,c_cont)
71 | %
72 | title('Sequência exponencial interpretada com fasor:
73 |           x[n] = e^{j (\Omega_0 n + \Theta_0)}')
74 | ylabel('cos(\Omega_0 n + \Theta_0)')
75 | zlabel('sin(\Omega_0 n + \Theta_0)')
76 | xlabel('n')
77 | %
78 | grid on
79 | axis square
80 | view(8,12)
81 |
82 | %
83 | % EOF
84 | %

```

Código 4.3: Demonstração de cosseno.

```

1 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 | %
3 | % Arquivo: cos_demo.m
4 | %
5 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 |
7 | %
8 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9 | %
10 | % Titulo: Demo de sequencia senoidal
11 | %
12 | % Autor : Alexandre Santos de la Vega
13 | % Data : 30/03/2k9 ; 03/09/2k9
14 | %
15 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16 | %
17 | %
18 | % limpeza do ambiente de trabalho
19 | % variaveis
20 | % janelas
21 | clear all
22 | % janelas
23 | close all
24 |
25 | % definicao dos parametros

```

```

27 | n = 0:50;
| A = 1;
29 | % 1o quadrante
| Omega11 = (0); % ( = 2*pi ) (16 * Omega12) ( 8 * Omega12) ( 4 * Omega14)
31 | Omega12 = (2*pi)/16;
| Omega13 = (2*pi)/8; % ( 2 * Omega12)
33 | Omega14 = (2*pi)/4; % ( 2 * Omega13)
| % 2o quadrante
35 | Omega21 = (2*pi)/4;
| Omega22 = (2*pi)/(8/3); % ( 3 * Omega13)
37 | Omega23 = (2*pi)/(16/7); % ( 7 * Omega12)
| Omega24 = (2*pi)/2;
39 | % 3o quadrante
| Omega31 = (2*pi)/2;
41 | Omega32 = (2*pi)/(16/9); % ( 9 * Omega12)
| Omega33 = (2*pi)/(8/5); % ( 5 * Omega13)
43 | Omega34 = (2*pi)/(4/3);
| % 4o quadrante
45 | Omega41 = (2*pi)/(4/3);
| Omega42 = (2*pi)/(8/7); % ( 7 * Omega13)
47 | Omega43 = (2*pi)/(16/15); % (15 * Omega12)
| Omega44 = (2*pi);
49
FigCntr = 0;
51
53 % grafico de angulos OmegaXY
k=1;
55 z=[];
p=[

57 | cos(Omega11) + j*sin(Omega11), cos(Omega11) - j*sin(Omega11), ...
| cos(Omega12) + j*sin(Omega12), cos(Omega12) - j*sin(Omega12), ...
59 | cos(Omega13) + j*sin(Omega13), cos(Omega13) - j*sin(Omega13), ...
| cos(Omega14) + j*sin(Omega14), cos(Omega14) - j*sin(Omega14), ...
61 | cos(Omega21) + j*sin(Omega21), cos(Omega21) - j*sin(Omega21), ...
| cos(Omega22) + j*sin(Omega22), cos(Omega22) - j*sin(Omega22), ...
63 | cos(Omega23) + j*sin(Omega23), cos(Omega23) - j*sin(Omega23), ...
| cos(Omega24) + j*sin(Omega24), cos(Omega24) - j*sin(Omega24), ...
65 | cos(Omega31) + j*sin(Omega31), cos(Omega31) - j*sin(Omega31), ...
| cos(Omega32) + j*sin(Omega32), cos(Omega32) - j*sin(Omega32), ...
67 | cos(Omega33) + j*sin(Omega33), cos(Omega33) - j*sin(Omega33), ...
| cos(Omega34) + j*sin(Omega34), cos(Omega34) - j*sin(Omega34), ...
69 | cos(Omega41) + j*sin(Omega41), cos(Omega41) - j*sin(Omega41), ...
| cos(Omega42) + j*sin(Omega42), cos(Omega42) - j*sin(Omega42), ...
71 | cos(Omega43) + j*sin(Omega43), cos(Omega43) - j*sin(Omega43), ...
| cos(Omega44) + j*sin(Omega44), cos(Omega44) - j*sin(Omega44), ...
73 ];

75 FigCntr = (FigCntr + 1);
figure(FigCntr)
77 pzmap(zpk(z,p,k))
title('Conjunto de ângulos utilizados no exemplo');

81 % graficos

83 % 1o quadrante
FigCntr = (FigCntr + 1);
85 figure(FigCntr)

```

```

87 %  

87 subplot(4,1,1)  

88 stem(n, (A*cos(Omega11*n)) )  

89 %  

90 hold on  

91 x = (0:2);  

92 y = (A*cos(Omega11*x));  

93 stem(x, y, 'y')  

94 hold off  

95 %  

96 ylabel('Omega = 0')  

97 title('Sinal senoidal: A*cos( \Omega n), \Omega in [0, \pi / 2] ')  

98  

99 subplot(4,1,2)  

100 stem(n, (A*cos(Omega12*n)) )  

101 %  

102 hold on  

103 na = 0:.1:50;  

104 plot(na, (A*cos(Omega12*na)), 'k')  

105 hold off  

106 %  

107 hold on  

108 x = (0:2:25);  

109 y = (A*cos(Omega12*x));  

110 stem(x, y, 'r')  

111 hold off  

112 %  

113 ylabel('Omega = (2*pi)/16')  

114  

115 subplot(4,1,3)  

116 stem(n, (A*cos(Omega13*n)) )  

117 %  

118 hold on  

119 na = 0:.1:50;  

120 plot(na, (A*cos(Omega13*na)), 'k')  

121 hold off  

122 %  

123 hold on  

124 x = (0:(length(y)-1));  

125 stem(x, y, 'r')  

126 hold off  

127 %  

128 hold on  

129 x = (32:2:50);  

130 y = (A*cos(Omega13*x));  

131 stem(x, y, 'g')  

132 hold off  

133 %  

134 ylabel('Omega = (2*pi)/8')  

135  

136 subplot(4,1,4)  

137 stem(n, (A*cos(Omega14*n)) )  

138 %  

139 hold on  

140 na = 0:.1:50;  

141 plot(na, (A*cos(Omega14*na)), 'k')  

142 hold off  

143 %  

144 hold on

```

```

145 | x = (16:(16+(length(y)-1)));
146 | stem(x, y, 'g')
147 | hold off
148 |
149 |
150 | hold on
151 | x = (0:4:8);
152 | y = (A*cos(Omega14*x));
153 | stem(x, y, 'y')
154 | hold off
155 |
156 | ylabel('Omega = (2*pi)/4')
157

158 |
159 | % 2o quadrante
160 | FigCntr = (FigCntr + 1);
161 | figure(FigCntr)
162 |
163 | subplot(4,1,1)
164 | stem(n, (A*cos(Omega21*n)) )
165 |
166 | hold on
167 | na = 0:.1:50;
168 | plot(na, (A*cos(Omega21*na)), 'k')
169 | hold off
170 |
171 | ylabel('Omega = (2*pi)/4')
172 | title('Sinal senoidal: A*cos( \Omega n), \Omega \in [\pi / 2 , \pi ]')
173

174 | subplot(4,1,2)
175 | stem(n, (A*cos(Omega22*n)) )
176 |
177 | hold on
178 | na = 0:.1:50;
179 | plot(na, (A*cos(Omega22*na)), 'k')
180 | hold off
181 |
182 | ylabel('Omega = (2*pi)/(8/3)')
183

184 | subplot(4,1,3)
185 | stem(n, (A*cos(Omega23*n)) )
186 |
187 | hold on
188 | na = 0:.1:50;
189 | plot(na, (A*cos(Omega23*na)), 'k')
190 | hold off
191 |
192 | ylabel('Omega = (2*pi)/(16/7)')
193

194 | subplot(4,1,4)
195 | stem(n, (A*cos(Omega24*n)) )
196 |
197 | hold on
198 | na = 0:.1:50;
199 | plot(na, (A*cos(Omega24*na)), 'k')
200 | hold off
201 |
202 | ylabel('Omega = (2*pi)/2')
203

```

```

205 % 3o quadrante
206 FigCntr = (FigCntr + 1);
207 figure(FigCntr)
208 %
209 subplot(4,1,1)
210 stem(n, (A*cos(Omega31*n)) )
211 %
212 ylabel(' \Omega = (2*pi)/2 ')
213 title(' Sinal senoidal: A*cos( \Omega n), \Omega \in [\pi , 3 \pi / 2 ] ')
214
215 subplot(4,1,2)
216 stem(n, (A*cos(Omega32*n)) )
217 %
218 ylabel(' \Omega = (2*pi)/(16/9) ')
219
220 subplot(4,1,3)
221 stem(n, (A*cos(Omega33*n)) )
222 %
223 ylabel(' \Omega = (2*pi)/(8/5) ')
224
225 subplot(4,1,4)
226 stem(n, (A*cos(Omega34*n)) )
227 %
228 ylabel(' \Omega = (2*pi)/(4/3) ')
229

230 %
231 % 4o quadrante
232 FigCntr = (FigCntr + 1);
233 figure(FigCntr)
234 %
235 subplot(4,1,1)
236 stem(n, (A*cos(Omega41*n)) )
237 %
238 ylabel(' \Omega = (2*pi)/(4/3) ')
239 title(' Sinal senoidal: A*cos( \Omega n), \Omega \in [3 \pi / 2 , 2 \pi ] ')
240
241 subplot(4,1,2)
242 stem(n, (A*cos(Omega42*n)) )
243 %
244 ylabel(' \Omega = (2*pi)/1(8/7) ')
245
246 subplot(4,1,3)
247 stem(n, (A*cos(Omega43*n)) )
248 %
249 ylabel(' \Omega = (2*pi)/(16/15) ')
250
251 subplot(4,1,4)
252 stem(n, (A*cos(Omega44*n)) )
253 %
254 ylabel(' \Omega = (2*pi) ')
255

256 %
257 % comparacoes
258 FigCntr = (FigCntr + 1);
259 figure(FigCntr)
260 %
261 subplot(4,1,1)
262 stem(n, (A*cos(Omega12*n)) )

```

```

263  %
264  hold on
265  x = (0:7:50);
266  y = (A*cos(Omega12*x));
267  stem(x, y, 'r')
268  hold off
269  %
270  hold on
271  x = (0:9:50);
272  z = (A*cos(Omega12*x));
273  stem(x, z, 'g')
274  hold off
275  %
276  hold on
277  x = (0:15:50);
278  w = (A*cos(Omega12*x));
279  stem(x, w, 'y')
280  hold off
281  %
282  ylabel('\'Omega = (2*pi)/16')
283  title('Sinal senoidal: A*cos( \'Omega n )')

285  subplot(4,1,2)
286  stem(n, (A*cos(Omega23*n)) )
287  %
288  hold on
289  x = (0:(length(y)-1));
290  stem(x, y, 'r')
291  hold off
292  %
293  ylabel('\'Omega = (2*pi)/(16/7)')

295  subplot(4,1,3)
296  stem(n, (A*cos(Omega32*n)) )
297  %
298  hold on
299  x = (0:(length(z)-1));
300  stem(x, z, 'g')
301  hold off
302  %
303  ylabel('\'Omega = (2*pi)/(16/9)')

305  subplot(4,1,4)
306  stem(n, (A*cos(Omega43*n)) )
307  %
308  hold on
309  x = (0:(length(w)-1));
310  stem(x, w, 'y')
311  hold off
312  %
313  ylabel('\'Omega = (2*pi)/(16/15)')

315  %
316  % EOF
317  %

```

Código 4.4: Demonstração da periodicidade de cosseno discreto.

```

2 %
% Arquivo: cos_periodicidade_discreto.m
4 %
%
6 %

8 %
%
10 %
% Titulo: Demo de
12 %         periodicidade de cos( . )
%         discreto
14 %
% Autor : Alexandre Santos de la Vega
16 % Data  : 22/10/2024
%
18 %
%
20 %
%
22 %
% limpeza do ambiente de trabalho
24 %
% variaveis
clear all
26 %
% janelas
close all
28 %
%
30 %
%
32 % - Periodicidade: (Omega * Np) = (K * 2*pi)
%
34 % Período: Np = K * (2*pi / Omega)
%
36 %
%
38 % - Para: Omega = 2*pi / N = 2*pi / (p/q) = (2*pi * q) / p
% tem-se: Np = K * (p/q)
%
40 % + Para: p e q inteiros
% tem-se:
42 %     * Período fundamental: Nf = p
%     * Quantidade de (2*pi) ciclos para Nf: #Ciclos = q
44 %

46 %
p1 = 22;
48 q1 = 1;
N1 = (p1/q1);
50 %
p2 = 11;
52 q2 = 3;
N2 = (p2/q2);
54 %
p3 = 3*pi;
56 q3 = 1;
N3 = (p3/q3);

```

```

58 %
60 Omega1 = (2*pi)/N1;
61 Omega2 = (2*pi)/N2;
62 Omega3 = (2*pi)/N3;

64 %%%%%%
66 %
67 n = 0:80;
68 %
69 x1 = cos(Omega1 * n);
70 x2 = cos(Omega2 * n);
71 x3 = cos(Omega3 * n);

74 %%%%%%
76 %%%%%%
78 %
79 FigNbr = 0;
80 %
81 Omega1_deg = (Omega1 * 180 / pi);
82 Omega2_deg = (Omega2 * 180 / pi);
83 Omega3_deg = (Omega3 * 180 / pi);

86 %%%%%%
88 %
89 FigNbr = (FigNbr +1);
90 %
91 %
92 subplot(3,1,1)
93 figure(FigNbr)
94 stem(n,x1)
95 ylabel('x_1 [n]')
96 %
97 tit_str = {
98     ['Condição de periodicidade: (\Omega * N_p) = (K * 2 \pi)',...,
99      ', ; , ...'],
100    ['Período: N_p = K * (2 \pi / \Omega) .'],
101    ['Para: ', ...],
102    '\Omega = 2 \pi / N = 2 \pi / (p/q) = (2 \pi * q) / p , , ...',
103    ['tem-se: ', ...],
104    ['N_p = K * (p/q) , N_f = p e Quant 2\pi\ciclos para N_f = q .'],
105    ['Sinal periódico: x_1 [n] = cos(\Omega_1 n)',...,
106      ', ; , ...'],
107    '\Omega_1 = 2 \pi / N_1 = , num2str(Omega1), , rad = , ...',
108    ['num2str(Omega1_deg), , graus', ...],
109    ', ; , ...'],
110    ['N_1 = p_1/q_1 = , num2str(p1), '/ , num2str(q1), ' .'],
111    '}';
112 title(tit_str)
113 %
114 %
115 subplot(3,1,2)
116 figure(FigNbr)

```

```

118 | stem(n,x2)
119 | ylabel('x_2 [n]')
120 | %
120 | tit_str = [
121 |   ['Sinal periódico: x_2 [n] = cos(\Omega_2 n)', ...]
122 |   ', ; , ...'
123 |   '\Omega_2 = 2 \pi / N_2 = ', num2str(Omega2), ' rad = ', ...
124 |   num2str(Omega2_deg), ' graus', ...
125 |   ', ; , ...'
126 |   'N_2 = p_2/q_2 = ', num2str(p2), '/ ', num2str(q2), ' . ']
127 | ];
128 | title(tit_str)
129 | %
130 | %
131 | subplot(3,1,3)
132 | figure(FigNbr)
133 | stem(n,x3)
134 | ylabel('x_3 [n]')
135 | %
136 | tit_str = [
137 |   ['Sinal NÃO periódico: x_3 [n] = cos(\Omega_3 n)', ...]
138 |   ', ; , ...'
139 |   '\Omega_3 = 2 \pi / N_3 = ', num2str(Omega3), ' rad = ', ...
140 |   num2str(Omega3_deg), ' graus', ...
141 |   ', ; , ...'
142 |   'N_3 = p_3/q_3 = ', num2str(p3), '/ ', num2str(q3), ' . ']
143 | ];
144 | title(tit_str)
145 | %
146 | xlabel('n')
147 | %
148 | %
149 | %
150 | %
151 | %
152 | %
153 | % EOF
154 | %

```

Código 4.5: Demonstração da periodicidade de cosseno amostrado.

```

2 %
% Arquivo: cos_periodicidade_amostrado.m
4 %
% %
6 %

8 %
% %
10 %
% Titulo: Demo de
12 %         periodicidade de cos( . )
13 %         amostrado
14 %
15 % Autor : Alexandre Santos de la Vega
16 % Data  : 30/03/2k9 ; 03/09/2k9

```

```

18 %  

19 % limpeza do ambiente de trabalho  

20 % variaveis  

21 clear all  

22 % janelas  

23 close all  

24  

25  

26  

27  

28 % definicao dos parametros  

29 ne = 0:1000;  

30 Femu = 500;  

31 Temu = 1/Femu;  

32  

33  

34 n = 0:30;  

35  

36 Fs1 = 24;  

37 Ts1 = 1/Fs1 ;  

38  

39 Fs2 = 25;  

40 Ts2 = 1/Fs2 ;  

41  

42 Fs3 = 10*sqrt(3);  

43 Ts3 = 1/Fs3 ;  

44  

45 f1 = 3;  

46 f2 = 3;  

47 f3 = 3;  

48  

49  

50 % calculo das sequencias  

51 x1a = cos (2*pi*f1*(ne*Temu));  

52 x2a = cos (2*pi*f2*(ne*Temu));  

53 x3a = cos (2*pi*f3*(ne*Temu));  

54  

55 x1 = cos (2*pi*f1*(n*Ts1));  

56 x2 = cos (2*pi*f2*(n*Ts2));  

57 x3 = cos (2*pi*f3*(n*Ts3));  

58  

59  

60 % graficos  

61 %  

62 % obtém tamanho de fullscreen  

63 scrsz = get(0,'ScreenSize'); %[left ,bottom ,width ,height]  

64 %  

65 % tenta evitar top bar e bottom bar  

66 % ajusta tipo de papel  

67 h1 = figure('PaperType','a4',  

68 'Position',[scrsz(1) (0.07 * scrsz(4)) scrsz(3) (0.81 * scrsz(4))]);  

69 %  

70 % graficos amostrados  

71 subplot(3,2,2)  

72 stem(n, x1)  

73 %  

74 hold on  

75 r = 0:8;

```

```

76 | stem(r, x1(r+1), 'r')
77 | hold off
78 |
79 | 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)
83 stem(n, x2)
84 %
85 hold on
86 r = 0:25;
87 stem(r, x2(r+1), 'r')
88 hold off
89 %
90 title ('\Omega = (\omega Ts) = (2 \pi f Ts)')
91 ylabel ('\Omega_2 = 6/25 \pi ; N_0 = 25')

92 subplot(3,2,6)
93 stem(n, x3)
94 %
95 hold on
96 r = 0:30;
97 stem(r, x3(r+1), 'r')
98 hold off
99 %
100 title ('N_f = K_f * (F_S / f_0)')
101 ylabel ('\Omega_3 = 6/(10 sqrt(3)) \pi ; Não periódico!')

102 xlabel('Amostra (n)')

106 %
107 % graficos analogicos
108 subplot(3,2,1)
109 plot((ne*Temu), x1a)
110 %
111 hold on
112 x = (0:8)*Ts1;
113 y = x1(1:length(x));
114 stem(x, y, 'r')
115 hold off
116 %
117 ylabel('F_{S_1} = 24 Hz')
118 title('Sinal analógico: cos(2 \pi f t) ; f_0 = 3 Hz')
119 grid on

120 subplot(3,2,3)
121 plot((ne*Temu), x2a)
122 %
123 hold on
124 x = (0:25)*Ts2;
125 y = x2(1:length(x));
126 stem(x, y, 'r')
127 hold off
128 %
129 ylabel('F_{S_2} = 25 Hz')
130 grid on

132 subplot(3,2,5)
133 plot((ne*Temu), x3a)
134

```

```
136 %  
136 hold on  
136 x = (0:30)*Ts3;  
138 y = x3(1:length(x));  
138 stem(x, y, 'r')  
140 hold off  
140 %  
142 ylabel('F_{S_3} = 10 \sqrt{3} Hz')  
142 xlabel('Tempo (s)')  
144 grid on  
146  
148 %  
148 % EOF  
150 %
```

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.

```

2 %
% Arquivo: amostragem.m
4 %
%
6

8 %
%
10 %
% Demos para %
12 % Apostila de DSP %
%
14 % Topico: processo de amostragem %
%
16 % Autor : Alexandre Santos de la Vega %
% Modifs: /2010-01/2011-02/
18 %
%
20 %

22 %
% limpa ambiente
24 clear all
25 close all
26

28 % define parametros
A0 = 1;
30 F0 = 1e3;
T0 = 1/F0;
32 %
NOP = 2;      % numero de periodos visualizados
34 %
PPPa = 1000; % numero de pontos por periodo para sinal analogico
36 Tsa = T0/PPPa;
t = 0:Tsa:(NOP*T0);
38 %
PPPs = 10;    % numero de pontos por periodo para sinal amostrado
40 Tss = T0/PPPs;
nTs = 0:Tss:(NOP*T0);
42 %

```

```

44    n      = 0:(length(nTs) - 1);

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

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

56    % desenha graficos
57    figure(1)
58    %
59    subplot(3,2,1)
60    plot(t,xa)
61    ylabel('x(t)')
62    xlabel('t (s)')
63    title('Sinal analógico: x(t)')
64    %
65    subplot(3,2,3)
66    for k = 1:length(nTs)
67        if (xs(k) == 0)
68            stem(nTs(k),xs(k),'k')
69        elseif (xs(k) > 0)
70            stem(nTs(k),xs(k),'k','^')
71        else
72            stem(nTs(k),xs(k),'k','v')
73        end
74        hold on
75    end
76    ylabel('x(t) * \delta_{T_S}(t)')
77    xlabel('t (s)')
78    title('Sinal multiplicado pelo trem de impulsos: x(t) * \delta_{T_S}(t)')
79    %
80    subplot(3,2,4)
81    stem(nTs,xs)
82    ylabel('x[n T_s]')
83    xlabel('t (s)')
84    title('Sinal amostrado: x[n T_s] = x(t), para t = n T_s')
85    %
86    subplot(3,2,2)
87    stem(nTs,ones(1,length(nTs)),'k','^')
88    ylabel('\delta_{T_S}(t)')
89    xlabel('t (s)')
90    title('Trem de impulsos: \delta_{T_S}(t)')
91    %
92    subplot(3,2,5)
93    %stem(nTs,[xs ; xsq]) % 'funciona, mas as cores nao ficam ok! ... '
94    stem(nTs,xs,'bo-')
95    hold on
96    stem(nTs,xsq,'ro-')
97    ylabel('[x[n T_S]]_Q')
98    xlabel('t (s)')
99    title('Sinal digital: [x[n T_S]]_Q = [x(t)]_Q, para t = n T_S')
100   %

```

```

102 | subplot(3,2,6)
103 | %stem(n,[xs ; xsq])' % 'funciona, mas as cores nao ficam ok! ...
104 | stem(n, xs, 'bo-')
105 | hold on
106 | stem(n, xsq, 'ro-')
107 | ylabel('[ x[n] ]_Q')
108 | xlabel('n')
109 | title('Sequência digital: [ x[n] ]_Q = [x[n T_S]]_Q')
110 |
111 |
112 | %
113 | % EOF
114 | %

```

Código 4.7: Demonstração de ambiguidade em amplitude e fase.

```

2 %
4 % Arquivo: ambiguidade_amp_fase_omega_pi.m
6 %
8 %
10 %
12 % Titulo: Demo de
13 %           superposicao de espectro
14 %           cos(pi*n + Theta0) = cos(pi * n)
16 %
18 %
20 %
22 % limpeza do ambiente de trabalho
23 % variaveis
24 clear all
25 % janelas
26 close all
28 %
29 % definicao dos parametros
30 ne = 0:500;
31 Femu = 500;
32 Temu = 1/Femu;
33 %
34 n = 0:15;
35 %
36 F0 = 3;
37 Theta0 = (2*pi)/8;
38 %
39 Fs = 2*F0;
40 Ts = 1/Fs;
41 %
42 % calculo das sequencias

```

```

42 | x1a = cos ( (2*pi*F0*(ne*Temu)) + Theta0 );
42 | x2a = cos ( (2*pi*F0*(ne*Temu)) ) * cos(Theta0);
44
44 | x1 = cos ( (2*pi*F0*(n*Ts)) + Theta0 );
46 | x2 = cos ( (2*pi*F0*(n*Ts)) ) * cos(Theta0);

48
48 % graficos
50 %
50 figure(1)
52 set(gcf, "papertype", "a4")
52 %
54 NORS = 6;
54 r = 0:NORS;
56 %
56 % graficos amostrados
58 subplot(2,2,2)
58 stem(n, x1)
58 %
58 hold on
62 stem(r, x1(r+1), 'r')
62 hold off
64 %
64 title('Sinais amostrados: x_1[n] = x_2[n]')
66 ylabel('x_1[n] = cos(pi n + Theta_0)')
AV = axis;
68 axis([AV(1) AV(2) (-1) (1)])

70 subplot(2,2,4)
70 stem(n, x2)
72 %
72 hold on
74 stem(r, x2(r+1), 'r')
74 hold off
76 %
76 title('\Omega = (\omega Ts) = (2 \pi f Ts) = \pi rad')
78 ylabel('x_2[n] = cos(pi n) * cos(Theta_0)')
78 xlabel('Amostra (n)')
80 AV = axis;
80 axis([AV(1) AV(2) (-1) (1)])
82

84 % graficos analogicos
84 %
86 x = (r)*Ts;
86 %
88 subplot(2,2,1)
88 plot((ne*Temu), x1a)
89 %
89 hold on
92 y = x1(1:length(x));
92 stem(x, y, 'r')
94 hold off
94 %
96 title('Sinais analógicos: x_1(t) \neq x_2(t)')
96 ylabel('x_1(t) = cos(2 \pi f_0 t + Theta_0)')
98 %
98 grid on
100

```

```

102 subplot(2,2,3)
103 plot((ne*Temu), x2a)
104 %
105 hold on
106 y = x2(1:length(x));
107 stem(x, y, 'r')
108 hold off
109 %
110 title('f_0 = 3 Hz ; F_S = (2 f_0) Hz ; \Theta_0 = (2 \pi) / 8 rad')
111 ylabel('x_2(t) = cos(2 \pi f_0 t) * cos(\Theta_0)')
112 xlabel('Tempo (s)')
113 AV = axis;
114 axis([AV(1) AV(2) (-1) (1)])
115 %
116 grid on
117 %
118
119 figure(2)
120 set(gcf, "papertype", "a4")
121 %
122 x = (r)*Ts;
123 %
124 y = x1(1:length(x));
125 stem(x, y, 'r')
126 hold on
127 plot((ne*Temu), x1a, 'k')
128 plot((ne*Temu), x2a, 'b')
129 %
130 tit_str_1 = [ 'Sinais analógicos: ', ...
131               'x_1(t) = cos(2 \pi f_0 t + \Theta_0) ', ...
132               '\neq cos(2 \pi f_0 t) * cos(\Theta_0) = x_2(t)' ];
133 tit_str_2 = [ 'f_0 = 3 Hz ; ', ...
134               'F_S = (2 f_0) Hz ; ', ...
135               '\Theta_0 = (2 \pi) / 8 rad ; ', ...
136               '\Omega = \pi rad' ];
137 tit_str_3 = [ 'Sinais amostrados: ', ...
138               'x_1[n] = cos (\pi n + \Theta_0) ', ...
139               '= cos(\pi n) * cos(\Theta_0) = x_2[n]' ];
140 title({tit_str_1, tit_str_2, tit_str_3})
141 ylabel('x_1(t) \neq x_2(t) ; x_1[n] = x_2[n]')
142 xlabel('Tempo (s)')
143 %
144 %grid on
145 %
146 %
147 % EOF
148 %

```

Código 4.8: Demonstração de *aliasing* - 1.

```

1 %
2 %
3 % Arquivo: aliasing_demo1.m
4 %
5 %

```

```

7
8 %
9 %%%
10 %
11 % Titulo : Demo de superposicao de espectro
12 %
13 % Autor : Alexandre Santos de la Vega
14 % Data : 30/03/2k9 ; 03/09/2k9
15 %
16 %%%
17 %

18 % limpeza do ambiente de trabalho
19 % variaveis
20 clear all
21 % janelas
22 close all

23

24

25 % definicao dos parametros
26 ne = 0:400;
27 %
28 Femu = 500;
29 Temu = 1/Femu;

30

31 n = 0:20;
32 %
33 f1 = 3;
34 f2 = 7;
35 f3 = 13;
36 %
37 Fs = 10;
38 Ts = 1/Fs;

39

40

41 % calculo das sequencias
42 x1a = cos (2*pi*f1*(ne*Temu));
43 x2a = cos (2*pi*f2*(ne*Temu));
44 x3a = cos (2*pi*f3*(ne*Temu));

45

46 x1 = cos (2*pi*f1*(n*Ts));
47 x2 = cos (2*pi*f2*(n*Ts));
48 x3 = cos (2*pi*f3*(n*Ts));

49

50

51 % graficos
52 %
53 figure(1)
54 set(gcf, "papertype", "a4")
55 %
56 NORS = 5;
57 r = 0:NORS;
58 %
59 % graficos amostrados
60 subplot(3,2,2)
61 stem(n, x1)
62 hold on
63 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 |
71 | subplot(3,2,4)  

| 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 |
81 | subplot(3,2,6)  

| 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 |
91 | xlabel( 'Amostra (n)')  

93 |
95 | % graficos analogicos  

| x = (r)*Ts;  

97 | %  

| subplot(3,2,1)  

| 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 |
109 | subplot(3,2,3)  

| 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 |
121 | subplot(3,2,5)  

| plot((ne*Temu), x3a)  

%  

123 | hold on

```

```

125 | x = (0:5)*Ts;
| y = x3(1:length(x));
| stem(x, y, 'r')
127 | hold off
| %
129 | ylabel('f_3 = 13 Hz')
| xlabel('Tempo (s)')
131 | %
133 | grid on
135 |
137 | figure(2)
138 | set(gcf, "papertype", "a4")
| %
139 | x = (r)*Ts;
| %
141 | y = x1(1:length(x));
| stem(x, y, 'r')
143 | hold on
| plot((ne*Temu), x1a, ':r')
145 | plot((ne*Temu), x2a, '-.b')
| plot((ne*Temu), x3a, 'k')
147 | %
148 | title({'Sinal analógico: cos(2 \pi f t)', ...
149 | 'Sinal amostrado: cos (\Omega n) = cos(2 \pi f n Ts)'})
150 | ylabel('f_1 = 3 Hz ; f_2 = 7 Hz ; f_3 = 13 Hz ; F_S = 10 Hz')
151 | xlabel('Tempo (s)')
| %
153 | %grid on
155 | %
156 | % EOF
157 | %

```

Código 4.9: Demonstração de *aliasing* - 2.

```

1 | %%%
2 | %
3 | % Arquivo: aliasing_demo2.m
4 | %
5 | %%%
7 |
9 | %%%
11 | % Titulo: Demo de superposicao de espectro
12 | %
13 | % Autor : Alexandre Santos de la Vega
14 | % Data : 30/03/2k9 ; 03/09/2k9
15 | %
17 | %
19 | % limpeza do ambiente de trabalho
20 | % variaveis

```

```

21 | clear all
22 | % janelas
23 | close all

25 |
26 | % definicao dos parametros
27 | ne = 0:400;
28 | %
29 | Femu = 500;
30 | Temu = 1/Femu;
31 |

33 | n = 0:20;

35 | Fs1 = 100;
36 | Ts1 = 1/Fs1;
37 |

39 | Fs2 = 10;
40 | Ts2 = 1/Fs2;

41 | Fs3 = 3/0.7;
42 | Ts3 = 1/Fs3;
43 |

45 | f1 = 3;
46 | f2 = 3;
47 | f3 = 3;

49 | % calculo das sequencias
50 | x1a = cos (2*pi*f1*(ne*Temu));
51 | x2a = cos (2*pi*f2*(ne*Temu));
52 | x3a = cos (2*pi*f3*(ne*Temu));
53 |

54 | x1 = cos (2*pi*f1*(n*Ts1));
55 | x2 = cos (2*pi*f2*(n*Ts2));
56 | x3 = cos (2*pi*f3*(n*Ts3));
57 |

59 | % graficos

61 | figure(1)
62 | set(gcf, "papertype", "a4")
63 | %
64 | %
65 | % graficos amostrados
66 | subplot(3,2,2)
67 | stem(n, x1)
68 | %
69 | hold on
70 | r = 0:11;
71 | stem(r, x1(r+1), 'r')
72 | hold off
73 | %
74 | 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)
78 | stem(n, x2)
79 | %

```

```

81 | hold on
81 | r = 0:7;
81 | stem(r, x2(r+1), 'r')
83 | hold off
83 | %
85 | ylabel('Omega_2 = (0.6 \pi) rad')
85 | title('Omega = (\omega Ts) = (2 \pi f Ts)')
87 |
87 | subplot(3,2,6)
89 | stem(n, x3)
89 | %
91 | hold on
91 | r = 0:3;
93 | stem(r, x3(r+1), 'r')
93 | hold off
95 | %
95 | ylabel('Omega_3 = 1.4 \pi = (2 \pi) - (0.6 \pi) rad')
97 | xlabel('Amostra (n)')
97 | %
99 | %
99 | % graficos analogicos
101 | subplot(3,2,1)
101 | plot((ne*Temu), x1a)
103 | %
103 | hold on
105 | x = (0:11)*Ts1;
105 | y = x1(1:length(x));
107 | stem(x, y, 'r')
107 | hold off
109 | %
109 | ylabel('F_{S_1} = 100 Hz')
111 | title('Sinal analógico: cos(2 \pi f t) ; f_0 = 3 Hz')
111 | grid on
113 |
113 | subplot(3,2,3)
115 | plot((ne*Temu), x2a)
115 | %
117 | hold on
117 | x = (0:7)*Ts2;
119 | y = x2(1:length(x));
119 | stem(x, y, 'r')
121 | hold off
121 | %
123 | ylabel('F_{S_2} = 10 Hz')
123 | grid on
125 |
125 | subplot(3,2,5)
127 | plot((ne*Temu), x3a)
127 | %
129 | hold on
129 | x = (0:3)*Ts3;
131 | y = x3(1:length(x));
131 | stem(x, y, 'r')
133 | hold off
133 | %
135 | ylabel('F_{S_3} = (3 / 0.7) Hz')
135 | xlabel('Tempo (s)')
137 | %
137 | grid on

```

```

139 |
141 | % EOF
141 | %

```

Código 4.10: Demonstração de *aliasing*, interagindo com o usuário.

```

2 %
% Arquivo: aliasing_ask_user.m
4 %
%
6

8 %
%
10 %
% Titulo:
12 % Demonstração de aliasing , interagindo com o usuário. %
%
14 % Autor : Alexandre Santos de la Vega %
15 % Data   : 01/09/2k8 %
16 %
%
18 %

20 %
22 clear all
close all

24 %

26 F1 = 250;
F2 = 1250;
28 F3 = 2250;

30 %

32 Tmin = 1/F3;
Tmax = 1/F1;

34 %

36 NbrPtsPer = 100;

38 %
TStepAna = Tmin/NbrPtsPer ;
40 ta = 0:TStepAna:2*Tmax;

42 %
xa1 = cos(2*pi*F1*ta);
44 xa2 = cos(2*pi*F2*ta);
xa3 = cos(2*pi*F3*ta);

46 %

48 %
50 disp(' ')
      disp(['Componentes do sinal com: ', ' , ...'])

```

```

52      'f_1 = 250 Hz, f_2 = 1250 Hz e f_3 = 2250 Hz.'])
53 disp(' ')
54 Fs = input('Frequênci a de amostragem (Hz): ');
55 Ts = 1/Fs;
56 nTs = 0:Ts:2*Tmax;
57 %
58 xd1 = cos(2*pi*F1*nTs);
59 xd2 = cos(2*pi*F2*nTs);
60 xd3 = cos(2*pi*F3*nTs);

62 %
63 FigCntr = 0;
64 n=0:(length(nTs)-1);
65 %
66 FigCntr = (FigCntr + 1);
67 figure(FigCntr)
68 set(gcf,'papertype',"a4")

72 subplot(3,2,1)
73 plot(ta,xa1)
74 ylabel('f_1 = 250 Hz')
75 title('x(t) = cos (2 \pi f t)')
76 subplot(3,2,3)
77 plot(ta,xa2)
78 ylabel('f_2 = 1250 Hz')
79 subplot(3,2,5)
80 plot(ta,xa3)
81 ylabel('f_3 = 2250 Hz')
82 xlabel('t (s)')

84 subplot(3,2,2)
85 stem(nTs,xd1)
86 title('x[n T_s] = cos (2 \pi f n T_s)')
87 subplot(3,2,4)
88 stem(nTs,xd2)
89 subplot(3,2,6)
90 stem(nTs,xd3)
91 xlabel('n T_s (s)')

93
94 %
95 FigCntr = (FigCntr + 1);
96 figure(FigCntr)
97 set(gcf,'papertype',"a4")

98 subplot(3,2,1)
99 stem(nTs,xd1)
100 title('x[n T_s] = cos (2 \pi f n T_s)')
101 subplot(3,2,3)
102 stem(nTs,xd2)
103 subplot(3,2,5)
104 stem(nTs,xd3)
105 xlabel('n T_s (s)')

107 subplot(3,2,2)
108 stem(n,xd1)

```

```

110 | title('x[n] = cos (\Omega n) ; \Omega = (2 \pi f T_s)')
111 | subplot(3,2,4)
112 | stem(n,xd2)
113 | subplot(3,2,6)
114 | stem(n,xd3)
115 | xlabel('n')
116 |
117 |
118 | %
119 | FigCntr = (FigCntr + 1);
120 | figure(FigCntr)
121 | set(gcf, "papertype", "a4")
122 |
123 | subplot(3,2,1)
124 | plot(ta,xa1)
125 | ylabel('f_1 = 250 Hz')
126 | title('x(t) = cos (2 \pi f t)')
127 | subplot(3,2,3)
128 | plot(ta,xa2)
129 | ylabel('f_2 = 1250 Hz')
130 | subplot(3,2,5)
131 | plot(ta,xa3)
132 | ylabel('f_3 = 2250 Hz')
133 | xlabel('t (s)')
134 |
135 | subplot(3,2,2)
136 | stem(n,xd1)
137 | title('x[n] = cos (\Omega n) ; \Omega = (2 \pi f T_s)')
138 | subplot(3,2,4)
139 | stem(n,xd2)
140 | subplot(3,2,6)
141 | stem(n,xd3)
142 | xlabel('n')
143 |
144 |
145 | %
146 | FigCntr = (FigCntr + 1);
147 | figure(FigCntr)
148 | set(gcf, "papertype", "a4")
149 |
150 | subplot(3,3,1)
151 | plot(ta,xa1)
152 | ylabel('f_1 = 250 Hz')
153 | title('x(t) = cos (2 \pi f t)')
154 | subplot(3,3,4)
155 | plot(ta,xa2)
156 | ylabel('f_2 = 1250 Hz')
157 | subplot(3,3,7)
158 | plot(ta,xa3)
159 | ylabel('f_3 = 2250 Hz')
160 | xlabel('t (s)')
161 |
162 | subplot(3,3,2)
163 | stem(nTs,xd1)
164 | title('x[n T_s] = cos (2 \pi f n T_s)')
165 | subplot(3,3,5)
166 | stem(nTs,xd2)
167 | subplot(3,3,8)
168 | stem(nTs,xd3)

```

```

170 xlabel('n T_s (s)')
171 n=0:(length(nTs)-1);
172 subplot(3,3,3)
173 stem(n,xd1)
174 title('x[n] = cos (\Omega n) ; \Omega = (2 \pi f T_s)')
175 subplot(3,3,6)
176 stem(n,xd2)
177 subplot(3,3,9)
178 stem(n,xd3)
179 xlabel('n')
180
181 %
182 % EOF
183 %
184

```

Código 4.11: Demonstração de cadeia de processamento com *aliasing*.

```

2 %% Arquivo: DSP_2018_1_TS1_Gabarito.m
4 %
6
8 %
10 % Título :
11 % Demonstração de cadeia de processamento com aliasing %
12 %
14 %
15 % Disciplina :
16 % Processamento Digital de Sinais. %
17 % Professor : %
18 % Alexandre Santos de la Vega. %
19 % Período : %
20 % 2018-1. %
21 %
22 % Atividade : %
23 % Teste Surpresa 1 - 2018_1 - Gabarito. %
24 %
25 % Assunto : %
26 % Cadeia de processamento com amostragem inadequada , %
27 % causando aliasing , formada pelas seguintes etapas: %
28 %
29 % Mixagem Analogica --> Amostragem --> Tx --> ...
30 % ... --> Rx --> Interpolacao --> Filtragem. %
31 %
32 %
33 %
34 %
35 %
36 % Limpa ambiente
37

```

```

40 clear all
41 close all
42 %%%%%%
43 %%%%%%
44 %%%%%%
45 %%%%%%
46 % Parametros gerais
47 %
48 % Discretos
49 %
50 Fs = 44e3;
51 %
52 nd = 0:43;
53 nTs = nd/Fs;
54 %
55 % Analogicos
56 %
57 Fsa = 10*Fs;
58 %
59 n = 0:439;
60 t = n/Fsa;
61 %
62 %%%%%%
63 %
64 %
65 %%%%%%
66 %
67 % Sinais Tx
68 %
69 % Parametros
70 %
71 %
72 A1 = 1;
73 A2 = 1;
74 A3 = 1;
75 A4 = 1;
76 %
77 %
78 f1 = 2e3;
79 f2 = 21e3;
80 f3 = 24e3;
81 f4 = 40e3;
82 %
83 %
84 %
85 %
86 % Sinais analogicos
87 %
88 %
89 x1 = A1*cos(2*pi*f1*n/Fsa);
90 x2 = A2*cos(2*pi*f2*n/Fsa);
91 x3 = A3*cos(2*pi*f3*n/Fsa);
92 x4 = A4*cos(2*pi*f4*n/Fsa);
93 %
94 %
95 xlow = x1;
96 xmed = x2 + x3;
97 xhigh = x4;

```

```

98 %
100 x = xlow + xmed + xhigh;

102 %
103 % Sinais discretos Tx
104 %
105 %
106 x1d = A1*cos(2*pi*f1*nd/Fs);
107 x2d = A2*cos(2*pi*f2*nd/Fs);
108 x3d = A3*cos(2*pi*f3*nd/Fs);
109 x4d = A4*cos(2*pi*f4*nd/Fs);

110 %
111 %
112 xlowd = x1d;
113 xmedd = x2d + x3d;
114 xhighd = x4d;

115 %
116 xd = xlowd + xmedd + xhighd;

117 %%%%%%
118 %%%%%%
119 %%%%%%
120 %%%%%%
121 %%%%%%
122 %%%%%%
123 %%%%%%
124 % Sinais Rx

126 %
127 % Parametros
128 %
129 %
130 A5 = 1;
131 A6 = 1;

132 %
133 %
134 f5 = 4e3;
135 f6 = 20e3;

136 %

137 % Sinais analogicos

138 %

139 %
140 y1 = A1*cos(2*pi*f1*n/Fsa);
141 y2 = A2*cos(2*pi*f2*n/Fsa);
142 y5 = A5*cos(2*pi*f5*n/Fsa);
143 y6 = A6*cos(2*pi*f6*n/Fsa);
144 y7 = 0*n;

145 %

146 %
147 ylow = y1 + y5;
148 ymed = y2 + y6;
149 yhigh = y7;

150 %

151 %
152 y = ylow + ymed + yhigh;

153 %%%%%%
154 %%%%%%
155 %%%%%%
156 %

```

```

158 %%%%%%
160 % Graficos
162 %%%%%%
164 %
166 % Nota :
167 %
168 % Foram inseridos comandos de "return" para teste ...
169 %
170 %%%%%%
172 % Controle da numeracao dos graficos
174 %
176 FigNbr = 0;
178 %%%%%%
180 %
182 % Limites de abscissas
184 min_t = min(t);
185 max_t = max(t);
186 %
187 min_nTs = min(nTs);
188 max_nTs = max(nTs);
189 %
190 min_n = min(nd);
191 max_n = max(nd);
192 %

194 % Limites de ordenadas
196 %
197 min_amp = min(x);
198 max_amp = max(x);
199 %
200 %%%%%%
202 %
203 FigNbr = FigNbr+1;
204 figure(FigNbr);
205 set(gcf, "papertype", "a4")
206 %
207 subplot(4,3,1)
208 plot(t,x1)
209 hold on
210 stem(nd/Fs,x1d)
211 v = axis;
212 v(1) = min_t;
213 v(2) = max_t;
214 %v(3) = min_amp;
215 %v(4) = max_amp;

```

```

216 | axis(v)
217 | title('A_1 cos(2 \pi f_1 t) ; f_1 = 2 kHz')
218 | ylabel('x_1 (t)')
219 | %
220 | subplot(4,3,4)
221 | plot(t,x2)
222 | hold on
223 | stem(nd/Fs,x2d)
224 | v = axis;
225 | v(1) = min_t;
226 | v(2) = max_t;
227 | %v(3) = min_amp;
228 | %v(4) = max_amp;
229 | axis(v)
230 | title('A_2 cos(2 \pi f_2 t) ; f_2 = 21 kHz')
231 | ylabel('x_2 (t)')
232 | %
233 | subplot(4,3,7)
234 | plot(t,x3)
235 | hold on
236 | stem(nd/Fs,x3d)
237 | v = axis;
238 | v(1) = min_t;
239 | v(2) = max_t;
240 | %v(3) = min_amp;
241 | %v(4) = max_amp;
242 | axis(v)
243 | title('A_3 cos(2 \pi f_3 t) ; f_3 = 24 kHz')
244 | ylabel('x_3 (t)')
245 | %
246 | subplot(4,3,10)
247 | plot(t,x4)
248 | hold on
249 | stem(nd/Fs,x4d)
250 | v = axis;
251 | v(1) = min_t;
252 | v(2) = max_t;
253 | %v(3) = min_amp;
254 | %v(4) = max_amp;
255 | axis(v)
256 | title('A_4 cos(2 \pi f_4 t) ; f_4 = 40 kHz')
257 | ylabel('x_4 (t)')
258 | %
259 | xlabel('t')
260 | %
261 | subplot(4,3,2)
262 | stem(nd/Fs,x1d,'r')
263 | v = axis;
264 | v(1) = min_t;
265 | v(2) = max_t;
266 | %v(3) = min_amp;
267 | %v(4) = max_amp;
268 | axis(v)
269 | title('A_1 cos(2 \pi f_1 n T_S) ; F_S = 44 kHz')
270 | ylabel('x_1 (n T_S)')
271 | %
272 | subplot(4,3,5)
273 | stem(nd/Fs,x2d,'r')
274 | v = axis;

```

```

276 | v(1) = min_t;
276 | v(2) = max_t;
276 | %v(3) = min_amp;
278 | %v(4) = max_amp;
278 | axis(v)
280 | title('A_2 cos(2 \pi f_2 n T_S) ; F_S = 44 kHz')
280 | ylabel('x_2 (n T_S)')
282 |
282 | subplot(4,3,8)
284 | stem(nd/Fs,x3d,'r')
284 | v = axis;
286 | v(1) = min_t;
286 | v(2) = max_t;
288 | %v(3) = min_amp;
288 | %v(4) = max_amp;
290 | axis(v)
290 | title('A_3 cos(2 \pi f_3 n T_S) ; F_S = 44 kHz')
292 | ylabel('x_3 (n T_S)')
292 | %
294 | subplot(4,3,11)
294 | stem(nd/Fs,x4d,'r')
296 | v = axis;
296 | v(1) = min_t;
298 | v(2) = max_t;
298 | %v(3) = min_amp;
300 | %v(4) = max_amp;
300 | axis(v)
302 | title('A_4 cos(2 \pi f_4 n T_S) ; F_S = 44 kHz')
302 | ylabel('x_4 (n T_S)')
304 |
306 | xlabel('n T_S')
306 | %
308 | subplot(4,3,9)
308 | plot(t,y6)
308 | hold on
310 | stem(nd/Fs,x3d)
310 | v = axis;
312 | v(1) = min_t;
312 | v(2) = max_t;
314 | %v(3) = min_amp;
314 | %v(4) = max_amp;
316 | axis(v)
316 | title('A_6 cos(2 \pi f_6 t) ; f_6 = 20 kHz')
318 | ylabel('y_6 (t)')
318 | %
320 | subplot(4,3,12)
320 | plot(t,y5)
320 | hold on
322 | stem(nd/Fs,x4d)
322 | v = axis;
324 | v(1) = min_t;
324 | v(2) = max_t;
326 | %v(3) = min_amp;
326 | %v(4) = max_amp;
328 | axis(v)
330 | title('A_5 cos(2 \pi f_5 t) ; f_5 = 4 kHz')
330 | ylabel('y_5 (t)')
332 |
332 | xlabel('t')

```

```

334 %
335 % Insercao nao convencional de texto...
336 %
337 % Subplot vazio...
338 subplot(4,3,3)
339 axis("off")
340 % Definicao de cell array com strings de comprimentos diferentes...
341 title_str = {'* Sinais analógicos em azul.', ...
342     ' ', ' ', ...
343     '* Sinais discretos em vermelho.'} ;
344 % Insercao de texto no titulo de subplot vazio...
345 hdl = title(title_str, ...
346     "fontsize",14, ...
347     "horizontalalignment","left", ...
348     "verticalalignment","top");
349 title_position = get(hdl,"position");
350 title_position(1) = 0;
351 set(hdl,"position",title_position)
352 %
353 %return
354 %%%%%%
355 %
356 %
357 %
358 FigNbr = FigNbr+1;
359 figure(FigNbr);
360 set(gcf,"papertype","a4")
361 %
362 subplot(3,4,1)
363 plot(t,x1)
364 v = axis;
365 v(1) = min_t;
366 v(2) = max_t;
367 v(3) = min_amp;
368 v(4) = max_amp;
369 axis(v)
370 title('A_1 cos(2 \pi f_1 t) ; f_1 = 2 kHz')
371 ylabel('x_1 (t)')
372 %
373 xlabel('t')
374 %
375 subplot(3,4,2)
376 plot(t,x2)
377 v = axis;
378 v(1) = min_t;
379 v(2) = max_t;
380 v(3) = min_amp;
381 v(4) = max_amp;
382 axis(v)
383 title('A_2 cos(2 \pi f_2 t) ; f_2 = 21 kHz')
384 ylabel('x_2 (t)')
385 %
386 xlabel('t')
387 %
388 subplot(3,4,3)
389 plot(t,x3)
390 v = axis;
391 v(1) = min_t;
392 v(2) = max_t;

```

```

394 | v(3) = min_amp;
| v(4) = max_amp;
| axis(v)
396 | title('A_3 cos(2 \pi f_3 t) ; f_3 = 24 kHz')
| ylabel('x_3 (t)')
398 %
| xlabel('t')
400 %
401 subplot(3,4,4)
402 plot(t,x4)
v = axis;
404 v(1) = min_t;
405 v(2) = max_t;
406 v(3) = min_amp;
407 v(4) = max_amp;
408 axis(v)
409 title('A_4 cos(2 \pi f_4 t) ; f_4 = 40 kHz')
410 ylabel('x_4 (t)')
411 %
412 xlabel('t')
413 %
414 subplot(3,4,5)
415 stem(nd,x1d,'r')
416 v = axis;
417 v(1) = min_n;
418 v(2) = max_n;
419 v(3) = min_amp;
420 v(4) = max_amp;
axis(v)
421 title('x_1 [n] ; F_S = 44 kHz')
422 ylabel('x_1 [n]')
423 %
424 xlabel('n')
425 %
426 subplot(3,4,6)
427 stem(nd,x2d,'r')
v = axis;
429 v(1) = min_n;
430 v(2) = max_n;
431 v(3) = min_amp;
432 v(4) = max_amp;
433 axis(v)
434 title('x_2 [n] ; F_S = 44 kHz')
435 ylabel('x_2 [n]')
436 %
437 xlabel('n')
438 %
439 subplot(3,4,7)
440 stem(nd,x3d,'r')
441 v = axis;
442 v(1) = min_n;
443 v(2) = max_n;
444 v(3) = min_amp;
445 v(4) = max_amp;
axis(v)
446 title('x_3 [n] = y_6 [n] ; F_S = 44 kHz')
447 ylabel('x_3 [n]')
448 %
449 xlabel('n')

```

```
452 %  
453 subplot(3,4,8)  
454 stem(nd,x4d,'r')  
455 v = axis;  
456 v(1) = min_n;  
457 v(2) = max_n;  
458 v(3) = min_amp;  
459 v(4) = max_amp;  
460 axis(v)  
461 title('x_4 [n] = y_5 [n] ; F_S = 44 kHz')  
462 ylabel('x_4 [n]')  
463 %  
464 xlabel('n')  
465 %  
466 subplot(3,4,11)  
467 plot(t,y6)  
468 v = axis;  
469 v(1) = min_t;  
470 v(2) = max_t;  
471 v(3) = min_amp;  
472 v(4) = max_amp;  
473 axis(v)  
474 title('A_6 cos(2 \pi f_6 t) ; f_6 = 20 kHz')  
475 ylabel('y_6 (t)')  
476 %  
477 xlabel('t')  
478 %  
479 subplot(3,4,12)  
480 plot(t,y5)  
481 v = axis;  
482 v(1) = min_t;  
483 v(2) = max_t;  
484 v(3) = min_amp;  
485 v(4) = max_amp;  
486 axis(v)  
487 title('A_5 cos(2 \pi f_5 t) ; f_5 = 4 kHz')  
488 ylabel('y_5 (t)')  
489 %  
490 xlabel('t')  
491 %  
492 %return  
493 %%  
494 %%  
495 %  
496 FigNbr = FigNbr+1;  
497 figure(FigNbr);  
498 set(gcf,"papertype","a4")  
499 %  
500 subplot(3,4,1)  
501 plot(t,x1)  
502 v = axis;  
503 v(1) = min_t;  
504 v(2) = max_t;  
505 v(3) = min_amp;  
506 v(4) = max_amp;  
507 axis(v)  
508 title('A_1 cos(2 \pi f_1 t) ; f_1 = 2 kHz')  
509 ylabel('x_1 (t)')
```

```

512 %  

512 subplot(3,4,9)  

512 plot(t,xlow)  

514 v = axis;  

514 v(1) = min_t;  

516 v(2) = max_t;  

516 v(3) = min_amp;  

518 v(4) = max_amp;  

518 axis(v)  

520 title('Sinal Tx (low)')  

520 ylabel('x_{low} (t)')  

522 %  

522 xlabel('t')  

524 %  

524 subplot(3,4,2)  

526 plot(t,x2)  

526 v = axis;  

528 v(1) = min_t;  

528 v(2) = max_t;  

530 v(3) = min_amp;  

530 v(4) = max_amp;  

532 axis(v)  

532 title('A_2 cos(2 \pi f_2 t) ; f_2 = 21 kHz')  

534 ylabel('x_2 (t)')  

534 %  

536 subplot(3,4,6)  

536 plot(t,x3)  

538 v = axis;  

538 v(1) = min_t;  

540 v(2) = max_t;  

540 v(3) = min_amp;  

542 v(4) = max_amp;  

542 axis(v)  

544 title('A_3 cos(2 \pi f_3 t) ; f_3 = 24 kHz')  

544 ylabel('x_3 (t)')  

546 %  

546 subplot(3,4,10)  

548 plot(t,xmed)  

548 v = axis;  

550 v(1) = min_t;  

550 v(2) = max_t;  

552 v(3) = min_amp;  

552 v(4) = max_amp;  

554 axis(v)  

554 title('Sinal Tx (med)')  

556 ylabel('x_{med} (t)')  

556 %  

558 xlabel('t')  

558 %  

560 subplot(3,4,3)  

560 plot(t,x4)  

562 v = axis;  

562 v(1) = min_t;  

564 v(2) = max_t;  

564 v(3) = min_amp;  

566 v(4) = max_amp;  

566 axis(v)  

568 title('A_4 cos(2 \pi f_4 t) ; f_4 = 40 kHz')  

568 ylabel('x_4 (t)')

```

```
570 %  
571 subplot(3,4,11)  
572 plot(t,xhigh)  
573 v = axis;  
574 v(1) = min_t;  
575 v(2) = max_t;  
576 v(3) = min_amp;  
577 v(4) = max_amp;  
578 axis(v)  
579 title('Sinal Tx (high)')  
580 ylabel('x_{high}(t)')  
581 %  
582 xlabel('t')  
583 %  
584 subplot(3,4,12)  
585 plot(t,x)  
586 v = axis;  
587 v(1) = min_t;  
588 v(2) = max_t;  
589 v(3) = min_amp;  
590 v(4) = max_amp;  
591 axis(v)  
592 title('Sinal Tx (low + med + high)')  
593 ylabel('x(t)')  
594 %  
595 xlabel('t')  
596 %  
597 %return  
598 %%  
599 %%  
600 %  
601 FigNbr = FigNbr+1;  
602 figure(FigNbr);  
603 set(gcf,'papertype','a4')  
604 %  
605 subplot(4,1,1)  
606 plot(t,x,'b')  
607 v = axis;  
608 v(1) = min_t;  
609 v(2) = max_t;  
610 v(3) = min_amp;  
611 v(4) = max_amp;  
612 axis(v)  
613 title('Sinal Tx')  
614 ylabel('x(t)')  
615 %  
616 subplot(4,1,2)  
617 stem(nTs,xd,'r')  
618 v = axis;  
619 v(1) = min_nTs;  
620 v(2) = max_nTs;  
621 v(3) = min_amp;  
622 v(4) = max_amp;  
623 axis(v)  
624 title('Sinal Discreto Tx')  
625 ylabel('x(n T_s)')  
626 %  
627 subplot(4,1,3)
```

```

630 | plot(t,y,'k')
631 | v = axis;
632 | v(1) = min_t;
633 | v(2) = max_t;
634 | v(3) = min_amp;
635 | v(4) = max_amp;
636 | axis(v)
637 | title('Sinal Rx')
638 | ylabel('y (t)')
639 | %
640 | subplot(4,1,4)
641 | plot(t,x,'b')
642 | hold on
643 | stem(nTs,xd,'r')
644 | plot(t,y,'k')
645 | v = axis;
646 | v(1) = min_t;
647 | v(2) = max_t;
648 | v(3) = min_amp;
649 | v(4) = max_amp;
650 | axis(v)
651 | title('Superposição dos três sinais')
652 | %
653 | xlabel('t')
654 | %
655 | %return
656 | %%%%%%
657 |
658 | %
659 | FigNbr = FigNbr+1;
660 | figure(FigNbr);
661 | set(gcf,"papertype","a4")
662 | %
663 | subplot(3,4,2)
664 | plot(t,y1)
665 | v = axis;
666 | v(1) = min_t;
667 | v(2) = max_t;
668 | v(3) = min_amp;
669 | v(4) = max_amp;
670 | axis(v)
671 | title('A_1 cos(2 \pi f_1 t) ; f_1 = 2 kHz')
672 | ylabel('y_1 (t)')
673 | %
674 | subplot(3,4,3)
675 | plot(t,y2)
676 | v = axis;
677 | v(1) = min_t;
678 | v(2) = max_t;
679 | v(3) = min_amp;
680 | v(4) = max_amp;
681 | axis(v)
682 | title('A_2 cos(2 \pi f_2 t) ; f_2 = 21 kHz')
683 | ylabel('y_2 (t)')
684 | %
685 | subplot(3,4,10)
686 | plot(t,ylow)
687 | v = axis;

```

```
688 | v(1) = min_t;
689 | v(2) = max_t;
690 | v(3) = min_amp;
691 | v(4) = max_amp;
692 | axis(v)
693 | title('Sinal Rx (low)')
694 | ylabel('y_low (t)')
695 | %
696 | xlabel('t')
697 | %
698 | subplot(3,4,6)
699 | plot(t,y5)
700 | v = axis;
701 | v(1) = min_t;
702 | v(2) = max_t;
703 | v(3) = min_amp;
704 | v(4) = max_amp;
705 | axis(v)
706 | title('A_5 cos(2 \pi f_5 t) ; f_5 = 4 kHz')
707 | ylabel('y_5 (t)')
708 | %
709 | subplot(3,4,7)
710 | plot(t,y6)
711 | v = axis;
712 | v(1) = min_t;
713 | v(2) = max_t;
714 | v(3) = min_amp;
715 | v(4) = max_amp;
716 | axis(v)
717 | title('A_6 cos(2 \pi f_6 t) ; f_6 = 20 kHz')
718 | ylabel('y_6 (t)')
719 | %
720 | subplot(3,4,11)
721 | plot(t,ymed)
722 | v = axis;
723 | v(1) = min_t;
724 | v(2) = max_t;
725 | v(3) = min_amp;
726 | v(4) = max_amp;
727 | axis(v)
728 | title('Sinal Rx (med)')
729 | ylabel('y_med (t)')
730 | %
731 | xlabel('t')
732 | %
733 | subplot(3,4,4)
734 | plot(t,y7)
735 | v = axis;
736 | v(1) = min_t;
737 | v(2) = max_t;
738 | v(3) = min_amp;
739 | v(4) = max_amp;
740 | axis(v)
741 | title('Sinal nulo')
742 | ylabel('y_7 (t)')
743 | %
744 | subplot(3,4,12)
745 | plot(t,yhigh)
746 | v = axis;
```

```
748 | v(1) = min_t;
    | v(2) = max_t;
    | v(3) = min_amp;
    | 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;
    | axis(v)
764 | title('Sinal Rx (low + med + high)')
    | ylabel('y (t)')
    | %
    | 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.

```
1 %%  
2 %  
3 % Arquivo : DPZ_indica_mult_sing.m  
4 %  
5 %%  
6 %%  
7 %%  
8 %%  
9 %%  
10 %% DSP : Demo sobre tracado de DPZ,  
11 %% com sinalizacao de singularidades multiplas.  
12 %%  
13 %% Autor: Alexandre S. de la Vega.  
14 %%  
15 %% Datas: /2018-1/  
16 %%  
17 %%  
18 %%  
19 %%
```

```

21 %
23 % limpa workspace
24 clear all
25 close all

27 %
28 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
29 %

31 % calcula circulo de raio unitario
32 N = 360;
33 n = 0:(N-1);
34 unit_circ = exp(j*2*pi*n/N);
35 %
36 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
37 %
38 %

39 % define funcao polinomial racional
40 % para testes...
41 %

43 % numerador
44 %
45 num = [ 5 4 3 2 1 ];
46 %
47 num = [ 1 2 3 4 5 ];
48 %num = [ 1 2 3 4 ];
49 %num = [ 1 2 ];
50 %num = 1;
51 %
52 %num = [ 1 1 .25 ];
53 %
54 % erro numerico em roots()
55 % gera 2 zeros simples
56 % ao inves de 1 polo multiplo com m=2 ...
57 %num = [ 1.00000 2.50000 2.00000 0.50000 ];

59 % denominador
60 %
61 den = [ 5 4 3 2 1 ];
62 %den = [ 5 4 3 2 ];
63 %den = [ 5 4 3 ];
64 %den = [ 5 4 ];
65 %den = 5;
66 %
67 %den = [ 1 -1 .25 ];
68 %

69 %
70 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
71 %

73 nc = [ 1 .9 ];
74 np = [ 1 .8 ];
75 nf = [ 1 .5 ];
76 %

77 num = conv( [1 0] , conv(nc,np) );

```

```

79 | den = [ 1 0 0 0 ] + conv( conv(nc,np) , nf );
81 |
82 | %
83 | %
84 | % calcula zeros, polos e constante de ganho
85 | z = roots(num);
86 | p = roots(den);
87 | k = num(1)/den(1);
88 |
89 | %
90 | %
91 | %
92 | %
93 | %
94 | % define zeros e polos
95 | % para testes...
96 | %
97 | % zeros multiplos
98 | %z = [ .2 .3 .1 .2 .3 .3 .1 .2 ]';
99 | %
100 | % polos multiplos
101 | %p = [ -.3-.3*i -.3+.3*i , ...
102 | %           -.1-.1*i -.1+.1*i , ...
103 | %           -.2-.2*i -.2+.2*i , ...
104 | %           -.2-.2*i -.2+.2*i , ...
105 | %           -.3-.3*i -.3+.3*i , ...
106 | %           -.3-.3*i -.3+.3*i , ...
107 | %           -.3-.3*i -.3+.3*i , ...
108 | %           -.1-.1*i -.1+.1*i , ...
109 | %           -.2-.2*i -.2+.2*i , ...
110 | %           0]';

111 |
112 | %
113 | %
114 | %
115 | %
116 | % gerencia as singularidades simples e multiplas
117 |
118 | % zeros
119 |
120 | % seleciona os diferentes valores de zeros
121 z_unq = unique(z);

122 |
123 | % separa os zeros simples dos multiplos
124 | % e
125 | % identifica as multiplicidades
126 |
127 z_smp = [];
128 z_mlt = [];
129 z_mlt_m = [];

130 for ind = 1:length(z_unq)
131     unq_pos = find( z == z_unq(ind) );
132     m = length(unq_pos);
133     if ( m == 1)
134         z_smp = [ z_smp ; z_unq(ind) ];
135     else
136         z_mlt    = [ z_mlt      ; z_unq(ind) ];

```

```

139 z_mlt_m = [ z_mlt_m ; m ];
140 endif
141 endfor
142 %
143 clear unq_pos m
144 %
145 %      polos
146 %
147 %      seleciona os diferentes valores de polos
148 p_unq = unique(p);
149 %
150 %      separa os polos simples dos multiplos
151 %      e
152 %      identifica as multiplicidades
153 %
154 p_smp = [];
155 p_mlt = [];
156 p_mlt_m = [];
157 %
158 for ind = 1:length(p_unq)
159     unq_pos = find( p == p_unq(ind) );
160     m = length(unq_pos);
161     if ( m == 1)
162         p_smp = [ p_smp ; p_unq(ind) ];
163     else
164         p_mlt = [ p_mlt ; p_unq(ind) ];
165         p_mlt_m = [ p_mlt_m ; m ];
166     endif
167 endfor
168 %
169 clear unq_pos m
170 %
171 %%%%%%
172 %
173 %
174 %      cria grafico
175 %
176 %
177 %      calculos comuns a todos os graficos
178 %
179 re_z_unq = real(z_unq);
180 im_z_unq = imag(z_unq);
181 %
182 re_p_unq = real(p_unq);
183 im_p_unq = imag(p_unq);
184 %
185 mult_offset_x = .025;
186 mult_offset_y = .050;
187 %
188 %      calcula raio de visualizacao ,
189 %      baseado na posicao das singularidades (zeros e polos)
190 %
191 if ( isempty(z) && isempty(p) )
192     dpz_limit = 1;
193 else
194     max_re_sing = max( [max(abs(re_z_unq)) , max(abs(re_p_unq))] );
195     max_im_sing = max( [max(abs(im_z_unq)) , max(abs(im_p_unq))] );

```

```

197 |     max_sing      = max( [ max_re_sing      , max_im_sing      ] ) ;
198 |     dpz_limit = fix( max_sing) + 1;
199 | endif

201 | % cria canvas
202 | hf=figure(1);
203 | set(hf,'papertype','a4')

205 | % desenha circulo de raio unitario
206 | plot(real(unit_circ),imag(unit_circ),'b')
207 |
208 | % mantem todos os graficos na mesma figura
209 | hold on

211 | % desenha os zeros simples
212 | plot(real(z_smp),imag(z_smp), 'ok')

213 |
214 | % desenha os zeros multiplos
215 | plot(real(z_mlt),imag(z_mlt), 'ok')
216 | for ind = 1:length(z_mlt_m)
217 |     text( (real(z_mlt(ind)) + mult_offset_x) ,
218 |           (imag(z_mlt(ind)) + mult_offset_y) ,
219 |           num2str(z_mlt_m(ind)) )
220 | endfor

221 |
222 | % desenha os polos simples
223 | plot(real(p_smp),imag(p_smp), 'xk')

225 |
226 | % desenha os polos multiplos
227 | plot(real(p_mlt),imag(p_mlt), 'xk')
228 | for ind = 1:length(p_mlt_m)
229 |     text( (real(p_mlt(ind)) + mult_offset_x) ,
230 |           (imag(p_mlt(ind)) + mult_offset_y) ,
231 |           num2str(p_mlt_m(ind)) )
232 | endfor

233 |
234 | % escreve constante de ganho
235 | % em posicao definida pelos limites do grafico
236 | K_str = [ 'K = ', sprintf("%.3f",k)];
237 | K_str_offset = dpz_limit / 8;
238 | text( ((-dpz_limit) + K_str_offset) ,
239 |           (( dpz_limit) - K_str_offset) ,
240 |           K_str )
241 |
242 | % identifica o grafico
243 | title('Diagrama de polos e zeros (DPZ)')
244 | ylabel('Im\{\.\}')
245 | xlabel('Re\{\.\}')
246 |
247 | % habilita o grid na figura
248 | grid on

249 |
250 | % Controla formato do grafico
251 | % Deve vir ANTES do controle dos limites !!!
252 | axis('image')

253 |
254 | % controla limites
255 | v = [ (-dpz_limit) dpz_limit (-dpz_limit) dpz_limit ];

```

```
257 | %  
259 | %  
261 | %  
263 | % EOF
```

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.

```
1 |%%%%%  
2 |%  
3 |% Arquivo: dtfs_exm.m  
4 |%  
5 |%%%%%  
6 |  
7 |%  
8 |%%%%%  
9 |% DTFS example %  
10|%  
11|%  
12|% Author: Alexandre S. de la Vega %  
13|% Version: /2022_01_12/ %  
14|/2010_12_14/ %  
15|%  
16|%  
17|%  
18|%  
19|%%%%%
```

```

21 % clear workspace
22 clear all
23 close all

25 %%%%%%
26 %%%%%%
27 % define input sequence
28 %
29 x_fund_n = rand(1,10);
30 %
31 x_fund_n = 5:-1:-10;
32 x_fund_n = 10:-1:-5;
33 %
34 %x_fund_n = 10:-1:-10;
35 %x_fund_n = 10:-1:0;
36 %x_fund_n = 0:-1:-10;
37 %
38 N = length(x_fund_n);

41 %%%%%%
42 %%%%%%
43 % calculate DTFS coeficients
44 X_fund_k = zeros(1,N);
45 for k = 0:(N-1)
46     ind = k+1;
47     for n = 0:(N-1)
48         X_fund_k(ind) = X_fund_k(ind) + ...
49                         ( x_fund_n(n+1) * exp(j*k*((2*pi)/N)*n) );
50     end
51 end
52 X_fund_k = X_fund_k / N;
53 X_fund_k_mod = abs(X_fund_k);
54 X_fund_k_deg = (angle(X_fund_k)*180)/pi;

57 %
58 % draw sequences
59 %

61 %%%%%%
62 %%%%%%
63 %
64 FigNbr = 0;

67 %
68 bck_clr = [.1, .1, .1];
69 %

71 n = 0:(N-1);
72 k = 0:(N-1);

75 %
76 %abs_min = min(abs(x_fund_n));
77 abs_max = max(abs(x_fund_n));

79 %
80 %val_min = min(x_fund_n);

```

```

81    %val_max = max(x_fund_n);
82    %
83    x_per_n = [x_fund_n x_fund_n x_fund_n];
84    n_per   = 0:(length(x_per_n)-1);
85    %
86    X_per_k_mod = [X_fund_k_mod X_fund_k_mod X_fund_k_mod];
87    X_per_k_deg = [X_fund_k_deg X_fund_k_deg X_fund_k_deg];
88    k_per     = 0:(length(X_per_k_mod)-1);

89   %%%%%%%%%%%%%
90    %

91    %
92    FigNbr = FigNbr+1;
93    figure(FigNbr)
94    %
95    %
96    subplot(2,4,1)
97    stem(n_per, x_per_n)
98    axis([n_per(1) n_per(end) -abs_max abs_max])
99    set(gca, "color", bck_clr)
100   ylabel('x_{per} [n]')
101   xlabel('n')
102   title({'Periodic signal', 'x_{per} [n]'})
103   %
104   %

105   subplot(2,4,2)
106   stem(n, x_fund_n)
107   axis([n(1) n(end) -abs_max abs_max])
108   set(gca, "color", bck_clr)
109   ylabel('x_{fund} [n]')
110   xlabel('n')
111   title({'Representation of a periodic signal',
112          'Time (n)'})
113   %
114   %

115   subplot(2,4,3)
116   stem(k, X_fund_k_mod)
117   axis([k(1) k(end) 0 abs_max])
118   set(gca, "color", bck_clr)
119   ylabel('| X_{fund} [k] |')
120   title({'Representation of a periodic signal',
121          'Frequency (k)'})
122   %
123   %

124   subplot(2,4,7)
125   stem(k, X_fund_k_deg)
126   axis([k(1) k(end) -200 200])
127   set(gca, "color", bck_clr)
128   ylabel('angle X_{fund} [k] (degree)')
129   xlabel('k')
130   %
131   %

132   subplot(2,4,4)
133   stem(k_per, X_per_k_mod)
134   axis([k_per(1) k_per(end) 0 abs_max])
135   set(gca, "color", bck_clr)
136   ylabel('| X_{per} [k] |')
137   title({'Periodic signal', 'X_{per} [k]'})

```

```
139 %  
140 subplot(2,4,8)  
141 stem(k_per, X_per_k_deg)  
142 axis([k_per(1) k_per(end) -200 200])  
143 set(gca, "color", bck_clr)  
144 ylabel('\angle X_{per} [k] (degree)')  
145 xlabel('k')  
146 %%  
147 %% EOF  
148 %%  
149 %% EOF
```

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.

```

1  %% Arquivo: dtft_even_seq.m
2  %
3  % Arquivo: dtft_even_seq.m
4  %
5  %% Titulo: Demo de DTFT
6  %
7  % de seqüência com simetria par
8  %
9  % Autor : Alexandre Santos de la Vega
10 %
11 % Data   : / 2010_02 /
12 %
13 %
14 %
15 %
16 %
17 %
18 %
19 %

21 % Limpeza do ambiente
22 clear all
23 close all

25 %
26 n = -10:10;
27 x1n = (-2<=n) & (n<=2);
28 x2n = (0<=n) & (n<=4);
29

31 %
32 k = -1:.01:1;
33 W = pi*k;

35 %
36 X1W = ( 1 + cos(W) + cos(2*W) );
37 X1Wmod = abs(X1W);
38 X1Wang = angle(X1W);

39 X2W = ( 1 + cos(W) + cos(2*W) ) .* exp(-j*4*W);
40 X2Wmod = abs(X2W);
41 X2Wang = unwrap(angle(X2W));
42

45 %
46 figure(1)
47 %
48 subplot(3,2,1)
49 stem(n,x1n)
50 ylabel('x(n)')
51 xlabel('n')

```

```
53 | title('DTFT de sinal com simetria par')
53 | %
54 | subplot(3,2,3)
55 | plot(k,X1Wmod)
55 | ylabel('|X(\Omega)|')
56 | xlabel('Ω / π')
56 | %
57 | subplot(3,2,5)
58 | plot(k,X1Wang)
58 | ylabel('angle X(Ω)')
59 | xlabel('Ω / π')
60 |
61 | subplot(3,2,2)
62 | stem(n,x2n)
62 | ylabel('x(n)')
63 | xlabel('n')
63 | title('DTFT de sinal com simetria par')
63 | %
64 | subplot(3,2,4)
65 | plot(k,X2Wmod)
65 | ylabel('|X(\Omega)|')
66 | xlabel('Ω / π')
66 | %
67 | subplot(3,2,6)
68 | plot(k,X2Wang)
68 | ylabel('angle X(Ω)')
69 | xlabel('Ω / π')
70 |
71 |
72 |
73 |
74 |
75 |
76 |
77 |
78 |
79 |
80 |
81 |
82 |
83 | % EOF
83 | %
```

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.

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %
3 % Arquivo: dtfs_dtft_dft_gate_retangular.m
4 %
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7 %
8 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9 %
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11 % Titulo: Calculo de DTFS, DTFT e DFT %
12 % de uma sequencia gate retangular. %
13 %
14 % Autor : Alexandre Santos de la Vega %
15 % Data : / 2011_11_29 / %
16 %
17 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
18 %

19
20 % limpeza do ambiente de trabalho
21 % variaveis
22 clear all
23 % janelas
24 close all
25

26

27 % definicao dos parametros
28 Ng = 3;      % comprimento do gate = 2Ng + 1
29 %
30 Np = 20;     % periodo fundamental de repeticao
31 %
32 Nv = 2*Np;
33 n = -Nv:Nv; % janela de visualizacao
34 %
35

36

37 % definicao da sequencia nao periodica
38 xn = gate_retang_unitario(Ng,n);
39

40

41 % definicao da sequencia periodica
42 xp = gate_retang_unitario(Ng,n);
43 l=1;
44 while ((l*Np - Ng) < Nv),
45     xp = xp + gate_retang_unitario(Ng,(n + l*Np));
46     xp = xp + gate_retang_unitario(Ng,(n - l*Np));
47     l = l + 1;
48 end
49
50

```

```

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,
59 % pois sin(2*pi) ~= sin(0)
60 % 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,
71 % pois sin(2*pi) ~= sin(0)
72 % por erro de calculo numerico
73 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 krangle = (1:Np) + Nv;
xk(krangle) = Np * xkp(krangle);
82

84 %%%%%%
86 % ilustra resultados
88 %axis ([XMIN XMAX YMIN YMAX])
89
90 figure(1)
91 %
92 %subplot(2,3,1)
93 subplot(3,2,1)
stem(n,xp,'k')
94 ylabel('x_p[n]')
95 xlabel('n')
96 title('Sinal periódico')
97 %
98 %subplot(2,3,2)
99 subplot(3,2,3)
stem(n,xn,'k')
100 ylabel('x[n]')
101 xlabel('n')
102 title('Sinal não periódico')
103 %
104 %subplot(2,3,3)
105 subplot(3,2,5)
stem(n,xn,'k')
106 ylabel('x [n]')
107
108
109
110

```

```

112 xlabel( 'n' )
113 title( 'Sinal não periódico' )
114 %
115 %subplot(2,3,4)
116 subplot(3,2,2)
117 stem(k,xkp,'k')
118 ylabel( 'X_p[k]' )
119 xlabel( 'k' )
120 title( 'DTFS de x_p[n]' )
121 v = axis;
122 v(1) = min(k);
123 v(2) = max(k);
124 v(3) = min(xkp);
125 v(4) = max(xkp);
126 axis(v)
127 %
128 %subplot(2,3,5)
129 subplot(3,2,4)
130 plot(W,xW,'k')
131 ylabel( 'X(e^{j \Omega})' )
132 xlabel( '\Omega' )
133 title( 'DTFT de x[n]' )
134 v = axis;
135 v(1) = min(W);
136 v(2) = max(W);
137 v(3) = min(xW);
138 v(4) = max(xW);
139 axis(v)
140 %
141 %subplot(2,3,6)
142 subplot(3,2,6)
143 stem(k,xk,'k')
144 ylabel( 'X [k]' )
145 xlabel( 'k' )
146 title( 'N_p-point DFT de x [n]' )
147 v = axis;
148 v(1) = min(k);
149 v(2) = max(k);
150 v(3) = min(xk);
151 v(4) = max(xk);
152 axis(v)
153
154 %%%%%%
155
156 %
157 %
158 %% Para conferir o resultado da equacao
159 %% com o resultado da funcao fft(.)
160 %
161 %% fg = fft(xp(1:20));
162 %
163 %% figure(2)
164 %% stem(0:19, (abs(fg) .* real(exp(j*angle(fg))))) )
165 %% ylabel( 'X [k]' )
166 %% xlabel( 'k' )
167 %% title( 'N_p-point DFT de x [n], calculada pela função fft(x)' )
168 %% axis(v)
169 %%

```

```
170 |
172 | %%%%%%
174 | %
176 | % EOF
| %
```

6.5 DTFT \times 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.

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %
3  % Arquivo: dtft_interp_dft.m
4  %
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6  %
7  %
8  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9  % DTFT interpolated from DFT      %
10 % Author: Alexandre S. de la Vega %
11 % Version: /2010_12_14/           %
12 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13 %
14 %

15

16 % clear workspace
17 clear all
18 close all

19

20

21 % define input sequence
22 x_bas = rand(1,10);
23 N = length(x_bas);

24

25

26 % calculate DTFS coefficients
27 a = zeros(1,N);
28 for k = 0:(N-1)
29     ind = k+1;
30     for n = 0:(N-1)
31         a(ind) = a(ind) + ( x_bas(n+1) * exp(-j*k*((2*pi)/N)*n) );
32     end
33 end
34 a_mod = abs(a);
35 a_deg = (angle(a)*180)/pi;
36

37

38 % calculate DFT coefficients
39 X_k = fft(x_bas,N);
40 X_k_mod = abs(X_k);
41 X_k_deg = (angle(X_k)*180)/pi;
42

43

44 % calculate DTFT interpolation
45 m = 0:199;
46 M = length(m);
47 Omega_step = (2*pi)/M;
48 Omega = Omega_step .* m;
49

50 X_omega = zeros(1,M);

```

```

53 | for k = 0:(N-1)
54 |   sin_ratio_param = ( (N*Omega) - (k*2*pi) ) / 2;
55 |   sin_ratio = sin(sin_ratio_param) ./ sin(sin_ratio_param/N);
56 |   X_partial = a(k+1) .* sin_ratio .* ...
57 |     exp(-j.*((Omega-(k*((2*pi)/N))).*((N-1)/2)));
58 |   X_omega = X_omega + X_partial;
59 | end
60 | X_omega = X_omega / N;
61 | X_omega_mod = abs(X_omega);
62 | X_omega_deg = (angle(X_omega)*180)/pi;
63 |
64 |
65 | % draw sequences
66 | figure(1)
67 |
68 | n = 0:(N-1);
69 | k = 0:(N-1);
70 |
71 | subplot(4,2,1)
72 | stem(n, x_bas)
73 | ylabel('x[n]')
74 | xlabel('n')
75 | title('Representation of a nonperiodic signal (x[n]):'
76 |           'Time (n) X Frequency (k)')
77 |
78 | %x_per = [x_bas x_bas x_bas];
79 | %n_per = 0:(length(x_per)-1);
80 | %subplot(4,2,2)
81 | %stem(n_per, x_per)
82 | %ylabel('x_{per}[n]')
83 | %xlabel('n')
84 |
85 | subplot(4,2,3)
86 | stem(k, a_mod, 'r')
87 | ylabel('| a_{k} |')
88 | xlabel('k')
89 | title('DFT representation of a periodic extension (\Delta k = 2*\pi / N)')
90 |
91 | subplot(4,2,4)
92 | stem(k, a_deg, 'r')
93 | ylabel('|\angle a_{k}| (degree)')
94 | xlabel('k')
95 |
96 | subplot(4,2,5)
97 | stem(k, X_k_mod, 'r')
98 | ylabel('| X_{k} |')
99 | xlabel('k')
100 | title('FFT representation of a periodic extension (\Delta k = 2*\pi / N)')
101 |
102 | subplot(4,2,6)
103 | stem(k, X_k_deg, 'r')
104 | ylabel('|\angle X_{k}| (degree)')
105 | xlabel('k')
106 |
107 | subplot(4,2,7)
108 | plot(Omega, X_omega_mod)
109 | hold on
110 | stem(k*(2*pi/N),X_k_mod, 'r')

```

```
111 | ylabel('| X(e^{ j \Omega}) |')
112 | xlabel('Omega (rad)')
113 | title('Interpolated DTFT from the DFT representation (\Delta k = 2*\pi / N)')
114 | AV = axis;
115 | axis([AV(1) (2*pi) AV(3) AV(4)])
116 |
117 | subplot(4,2,8)
118 | plot(Omega, X_omega_deg)
119 | hold on
120 | stem(k*(2*pi/N),X_k_deg,'r')
121 | ylabel('angle X(e^{ j \Omega}) (degree)')
122 | xlabel('Omega (rad)')
123 | AV = axis;
124 | axis([AV(1) (2*pi) AV(3) AV(4)])
125 |
126 |
127 | %
128 | % 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.

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %
3 % Arquivo: dft_leakage_demo.m
4 %
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 %
7 %
8 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9 %
10 % Titulo: Demo de DFT de sequencia senoidal %
11 %
12 % Autor : Alexandre Santos de la Vega %
13 %
14 % Data : / 01/07/2k9 / 2010_12_14 /
15 %
16 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
17 %

18 %
19 % Limpeza do ambiente
20 clear all
21 close all
22

23

24 % Parametros da sequencia senoidal original
25 A0 = 2;
26 F0 = 100;
27 T0 = 1/F0;
28 P0 = 0;

29

30

31 % Parametros da amostragem
32 SamplingFactor = 128; % Numero de pontos por periodo
33 Fs = (SamplingFactor * F0); % Sampling frequency
34 Ts = 1/Fs;

35

36 Nper = 4;
37 Nppp = (T0/Ts); % Numero de pontos por periodo
38 Ntot = (Nper * Nppp); % Total de pontos amostrados

39

40

41 % Montagem da sequencia amostrada
42 n = 0:(Ntot-1);
43 x = A0*cos((2*pi*F0*(n*Ts)) + P0);

44

45

46 % Calculo da DFT sem leakage
47 NPfft = Ntot; % Numero de pontos da DFT sem leakage
48 k = 0:(NPfft-1);

49 X = fft(x,NPfft);

50 Xmod = abs(X);
51
52
53

```

```

55 Xang_rad = angle(X);
55 Xang_deg = Xang_rad*180/pi;

57
57 % Calculo da DFT com leakage, devido a nao casamento de periodos
59 NPfftlp = Ntot*((2*Nper)-1)/(2*Nper); % Numero de pontos da DFT com leakage
59 klp = 0:(NPfftlp-1);

61 Xlp = fft(x,NPfftlp);
63
63 Xlpmod = abs(Xlp);
65 Xlpang_rad = angle(Xlp);
65 Xlpang_deg = Xlpang_rad*180/pi;
67

69 % Calculo da DFT com leakage, devido a insercao de zeros
69 NPfftl = 3*Ntot; % Numero de pontos da DFT com leakage
71
71 n1 = 0:(NPfftl-1);
73 ZeroPaddingVector = zeros(1,(NPfftl-NPfft));
73 x1 = [x, ZeroPaddingVector];
75
75 k1 = 0:(NPfftl-1);
77 Xl = fft(x,NPfftl);

79 Xlmod = abs(Xl);
79 Xlang_rad = angle(Xl);
81 Xlang_deg = Xlang_rad*180/pi;

83
83 %
85 % Graficos
85 %
87
87 % Grafico da DFT com leakage, devido a insercao de zeros
89 figure(1)

91 subplot(2,2,1)
91 stem(n,x)
93 title('Sinal original: x[n] = A_0 cos(2\pi f_0 n T_s + \phi_0)')
93 xlabel('n')
95 ylabel('x[n]')
95 AV = axis;
97 axis([AV(1) NPfft AV(3) AV(4)]) 

99 subplot(2,2,2)
99 stem(k,Xmod)
101 title('Módulo da DFT de x[n]')
101 xlabel('k')
103 ylabel('| X[k] |')
103 AV = axis;
105 axis([AV(1) NPfft AV(3) AV(4)]) 

107 subplot(2,2,4)
107 stem(k,Xang_deg)
109 title('Ângulo de fase da DFT de x[n]')
109 xlabel('k')
111 ylabel('\angle X[k]')
111 AV = axis;

```

```

113 axis([AV(1) NPfft AV(3) AV(4)])
115
117 % Grafico da DFT com leakage, devido a nao casamento de periodos
118 figure(2)
119 subplot(2,2,1)
120 stem(klp,x(klp+1))
121 title('Sinal original: x[n] = A_0 cos(2\pi f_0 n T_s + \phi_0)')
122 xlabel('n')
123 ylabel('x[n]')
124 AV = axis;
125 axis([AV(1) NPfftlp AV(3) AV(4)])
126
127 subplot(2,2,2)
128 stem(klp,Xlpmod)
129 title('Módulo da DFT de x[n]')
130 xlabel('k')
131 ylabel('| X[k] |')
132 AV = axis;
133 axis([AV(1) NPfftlp AV(3) AV(4)])
134
135 subplot(2,2,4)
136 stem(klp,Xlpang_deg)
137 title('Ângulo de fase da DFT de x[n]')
138 xlabel('k')
139 ylabel('\angle X[k]')
140 AV = axis;
141 axis([AV(1) NPfftlp AV(3) AV(4)])
142
143 % Grafico da DFT com leakage, devido a insercao de zeros
144 figure(3)
145
146 subplot(2,2,1)
147 stem(nl,xl)
148 title('Sinal original: x[n] = A_0 cos(2\pi f_0 n T_s + \phi_0)')
149 xlabel('n')
150 ylabel('x[n]')
151 AV = axis;
152 axis([AV(1) NPfftl AV(3) AV(4)])
153
154 subplot(2,2,2)
155 stem(kl,Xlmod)
156 title('Módulo da DFT de x[n]')
157 xlabel('k')
158 ylabel('| X[k] |')
159 AV = axis;
160 axis([AV(1) NPfftl AV(3) AV(4)])
161
162 subplot(2,2,4)
163 stem(kl,Xlang_deg)
164 title('Ângulo de fase da DFT de x[n]')
165 xlabel('k')
166 ylabel('\angle X[k]')
167 AV = axis;
168 axis([AV(1) NPfftl AV(3) AV(4)])
169
170
171 %

```

173	% EOF
	%

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.

```

1 | %%%%%%
2 | %
3 | % Arquivo: cs_fft.m
4 | %
5 | %%%%%%
6 | %
7 | function [Gk,Hk] = cs_fft(g,h)
8 |
9 | %
10| % CS_FFT Complex sequence FFT
11| %     CS_FFT(g,h) returns G[k] and H[k] by using
12| %                     a single FFT operation.
13|
14|
15| x = g + (j*h);
16|
17| X_fft = fft(x);
18|
19| X_aux = [X_fft, X_fft(1)];
20| X_fft_conj_refl_circ = conj( flipr(X_aux(2:end)) );
21|
22| Gk = (X_fft + X_fft_conj_refl_circ) / (2);
23| Hk = (X_fft - X_fft_conj_refl_circ) / (2*j);
24|
25| end
26|
27| %
28| % EOF
29| %

```

Código 6.7: Exemplo de cálculo das N-DFTs de duas sequências reais através do cálculo da N-DFT de uma sequência complexa.

```

1 | %%%%%%
2 | %
3 | % Arquivo: two_seqs_complex_fft.m
4 | %
5 | %%%%%%

```

```

7 %
9 %
11 % Titulo: Calculo de FFT %
13 % de uma sequencia complexa, %
15 % formada por duas sequencias reais %
17 % distintas:  $x[n] = g[n] + j h[n]$  %
19 %
21 %

23 % limpeza do ambiente de trabalho
24 % variaveis
25 clear all
26 % janelas
27 close all

29 %
31 N = 40;
n = 0:(N-1);

33 %
35 g = zeros(1,N);
g(1:7) = 1;

37 h = zeros(1,N);
39 h(1:11) = 1;

41 %
43 tic
44 %
45 G_fft = fft(g);
H_fft = fft(h);
47 %
48 t_2r_fft = toc;

51 %
53 tic
54 %
55 [G_cs_fft, H_cs_fft] = cs_fft(g,h);
56 %
57 t_1c_fft = toc;

59 %

61 %
63 %AXIS([XMIN XMAX YMIN YMAX])

```

```

65 figure(1)
66 %
67 subplot(4,2,1)
68 stem(n,abs(G_fft), 'k')
69 ylabel('|G[k]|')
70 title('Módulos calculados por duas N-FFTs de sequências reais')
71 %
72 subplot(4,2,2)
73 stem(n, angle(G_fft), 'k')
74 ylabel('\angle G[k]')
75 title('Ângulos de fase calculados por duas N-FFTs de sequências reais')
76 %
77 subplot(4,2,3)
78 stem(n,abs(H_fft), 'k')
79 ylabel('|H[k]|')
80 %
81 subplot(4,2,4)
82 stem(n, angle(H_fft), 'k')
83 ylabel('\angle H[k]')
84 %
85 subplot(4,2,5)
86 stem(n,abs(G_cs_fft), 'r')
87 ylabel('|G[k]|')
88 title('Módulos calculados por uma N-FFT de sequência complexa')
89 %
90 subplot(4,2,6)
91 stem(n, angle(G_cs_fft), 'r')
92 ylabel('\angle G[k]')
93 title('Ângulos de fase calculados por uma N-FFT de sequência complexa')
94 %
95 subplot(4,2,7)
96 stem(n,abs(H_cs_fft), 'r')
97 ylabel('|H[k]|')
98 xlabel('k')
99 %
100 subplot(4,2,8)
101 stem(n, angle(H_fft), 'r')
102 ylabel('\angle H[k]')
103 xlabel('k')
104 %

105

106

107 figure(2)
108 %
109 subplot(2,2,1)
110 stem(n,abs(G_fft), 'k')
111 hold on
112 stem(n,abs(G_cs_fft), 'r')
113 ylabel('|G[k]|')
114 title('Comparação dos módulos')
115 %
116 subplot(2,2,2)
117 stem(n,angle(G_fft), 'k')
118 hold on
119 stem(n,angle(G_cs_fft), 'r')
120 ylabel('\angle G[k]')
121 title('Comparação dos ângulos de fase')
122 %
123 subplot(2,2,3)

```

```

125 | stem(n,abs(H_fft), 'k')
125 | hold on
126 | stem(n,abs(H_cs_fft), 'r')
127 | ylabel( '|H[k]|' )
128 | xlabel( 'k' )
129 |
130 | subplot(2,2,4)
131 | stem(n,angle(H_fft), 'k')
131 | hold on
132 | stem(n,angle(H_cs_fft), 'r')
133 | ylabel( '\angle H[k]' )
134 | xlabel( 'k' )

137 |
138 | %%%%%%
139 |
140 |
141 | %
141 | % EOF
142 | %
143 |

```

Código 6.8: Exemplo de cálculo da 2N-DFT de uma sequência real através do cálculo da N-DFT de uma sequência complexa.

```

1 | %%%%%%
1 | %
3 | % Arquivo: one_seq_complex_fft.m
3 | %
5 | %%%%%%
5 | %

7 |
7 | %
9 | %%%%%%
9 | %
9 | % Titulo: Calculo de FFT
9 | %%%%%%
11 | % de uma sequencia complexa,
11 | %%%%%%
13 | % formada por duas sequencias reais
13 | %%%%%%
13 | % provenientes de uma sequencia:
13 | %%%%%%
15 | % x[n] = g[n] + j h[n]
15 | %%%%%%
15 | % g[n] = v[2n]
15 | %%%%%%
17 | % h[n] = v[2n+1]
17 | %%%%%%
19 | % Autor : Alexandre Santos de la Vega
19 | %%%%%%
21 | % Data : / 2011_11_24 /
21 | %%%%%%
23 | %

25 |
25 | % limpeza do ambiente de trabalho
27 | % variaveis
27 | clear all
29 | % janelas
29 | close all
31 |
33 | % definicao dos parametros

```

```

35 | M = 40;
| m = 0:(M-1);
| ind_m = (m + 1);
37 | m_par = (2*m);
| m_impar = (2*m) + 1;
39 | ind_m_par = m_par + 1;
| ind_m_impar = m_impar + 1;
41 |
% N = 2*M;
43 n = 0:(N-1);

45
% definicao da sequencia de entrada
47 v = zeros(1,N);
v(1:11) = 1;

49
% calculo da FFT isoladamente
51 tic
%
53 V_fft = fft(v);
%
55 t_2r_fft = toc;

57
% calculo da FFT com sequencia complexa
59 tic
%
61 g = v(ind_m_par);
h = v(ind_m_impar);
%
63 [G_cs_fft, H_cs_fft] = cs_fft(g, h);
%
65 V_cs_fft(ind_m) = G_cs_fft + (exp(-j*(pi/M)*m) .* H_cs_fft);
67 V_cs_fft(ind_m + M) = G_cs_fft + (exp(-j*(pi/M)*(m+M)) .* H_cs_fft);
%
69 t_1c_fft = toc;

71 %%%%%%
73

75 % ilustra resultados

77 %AXIS([XMIN XMAX YMIN YMAX])

79 figure(1)
%
81 subplot(3,2,1)
stem(n, abs(V_fft), 'k')
ylabel('|V[k]|')
title('Módulo calculado por uma 2N-FFT de sequência real')
%
85 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')
%
89 subplot(3,2,3)
stem(n, abs(V_cs_fft), 'r')

```

```
93 | ylabel( '|V[k]|')
94 | title( 'Módulo calculado por uma N-FFT de sequência complexa')
95 |
96 | subplot(3,2,4)
97 | stem(n, angle(V_cs_fft), 'r')
98 | ylabel( '\angle V[k]')
99 | title( 'Ângulo de fase calculado por uma N-FFT de sequência complexa')
100 |
101 | subplot(3,2,5)
102 | stem(n, abs(V_fft), 'k')
103 | hold on
104 | stem(n, abs(V_cs_fft), 'r')
105 | ylabel( '|V[k]|')
106 | title( 'Comparaçāo dos módulos')
107 | xlabel( 'k')
108 |
109 | subplot(3,2,6)
110 | stem(n, angle(V_fft), 'k')
111 | hold on
112 | stem(n, angle(V_cs_fft), 'r')
113 | ylabel( '\angle V[k]')
114 | title( 'Comparaçāo dos ângulos de fase')
115 | xlabel( 'k')

116 |
117 | %%%%%%
118 |
119 |
120 | %
121 | % EOF
122 | %
```

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.

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %
3 % Arquivo: dft_fft_ops_comparison.m
4 %
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 %
7 %
8 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9 %
10 % Titulo: Comparacao do numero de operacoes %
11 % entre %
12 % DFT por equacao original %
13 % e %
14 % FFT implementada no simulador %
15 %
16 %
17 % Autor : Alexandre Santos de la Vega %
18 % Data : / 2011_11_17 /
19 %
20 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
21 %

22 %
23 % limpeza do ambiente de trabalho
24 % variaveis
25 clear all
26 % janelas
27 close all
28 %

29 %

30 %
31 % definicao dos parametros
N = 2:1500;
33 m = 1:10;
34 Np = 2.^m;
35 %

36 %
37 % calculo do numero de operacoes para DFT
mc_dft = N.^2;
39 ac_dft = N.*(N-1);
40 %
41 mcp_dft = mc_dft(Np);
acp_dft = ac_dft(Np);
42 %

43 %

44 %
45 % calculo do numero de operacoes para DFT
mc_fft = (N./2).*log2(N);
47 ac_fft = N.*log2(N);
48 %
49 mcp_fft = mc_fft(Np);
acp_fft = ac_fft(Np);
50 %

51 
```

```

53 % calculo da aceleracao
54 mc = mc_dft ./ mc_fft;
55 ac = ac_dft ./ ac_fft;
56 %
57 mcp = mc(Np);
58 acp = ac(Np);
59 %

61 %%%%%%
63 %
64 % ilustra resultados
65 figure(1)
66 %
67 %AXIS([XMIN XMAX YMIN YMAX])
68 max_mc_ac_fft = ceil(max(max(mc_fft),max(ac_fft)));
69 %
70 subplot(4,2,1)
71 stem(Np,mcp_dft,'k')
72 hold on
73 plot(N,mc_dft,'k')
74 ylabel('N^2')
75 title('Number of complex multiplications: DFT')
76 %
77 subplot(4,2,2)
78 stem(Np,acp_dft,'k')
79 hold on
80 plot(N,ac_dft,'k')
81 ylabel('N(N-1)')
82 title('Number of complex additions: DFT')
83 %
84 subplot(4,2,3)
85 stem(Np,mcp_fft,'r')
86 hold on
87 plot(N,mc_fft,'r')
88 v = AXIS;
89 v(4) = max_mc_ac_fft;
90 AXIS(v)
91 ylabel('N/2 log_2(N)')
92 title('Number of complex multiplications: FFT')
93 %
94 subplot(4,2,4)
95 stem(Np,acp_fft,'r')
96 hold on
97 plot(N,ac_fft,'r')
98 v = AXIS;
99 v(4) = max_mc_ac_fft;
100 AXIS(v)
101 ylabel('N log_2(N)')
102 title('Number of complex additions: FFT')
103 %
104 subplot(4,2,5)
105 stem(Np,mcp_dft,'k')
106 hold on
107 plot(N,mc_dft,'k')
108 %
109 stem(Np,mcp_fft,'r')
110 plot(N,mc_fft,'r')
111 title('Number of complex multiplications: DFT and FFT')

```

```
113 %  
113 subplot(4,2,6)  
113 stem(Np,acp_dft,'k')  
115 hold on  
115 plot(N,ac_dft,'k')  
117 %  
117 stem(Np,acp_fft,'r')  
119 plot(N,ac_fft,'r')  
119 title('Number of complex additions: DFT and FFT')  
121 %  
121 subplot(4,2,7)  
123 stem(Np,mcp,'b')  
123 hold on  
125 plot(N,mc,'b')  
125 title('Number-of-complex-multiplications gain')  
127 xlabel('N')  
127 %  
129 subplot(4,2,8)  
129 stem(Np,acp,'b')  
131 hold on  
131 plot(N,ac,'b')  
133 title('Number-of-complex-additions gain')  
133 xlabel('N')  
135  
137 %%%%%%  
139 %  
141 % EOF  
141 %
```

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 L^AT_EX and Electronic Publishing*. Addison-Wesley, Harlow, England, 4th edition, 2004.
- [MG04] F. Mittelbach and M. Goosens. *The L^AT_EX 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.