Exercícios para laboratório de Análise de Sinais Discretos

Prof. Dr. Marcelo de Oliveira Rosa

2 de julho de 2013

Capítulo 1

Programação de Funções

1. Crie uma função que ordene (de modo crescente ou decrescente, a critério do usuário desta) usando o método da bolha (ou "bubblesort"). O protótipo da função será:

```
[resultado] = ordena_bolha(vetor, ordem)
```

   O parâmetro `ordem` indica em que ordem `vetor` será ordenado. Se `ordem`$= 0$ então a ordem será decrescente. Caso contrário, a ordem será crescente.

2. As funções `tic` e `toc` (do Matlab) são usadas para cronometrar o tempo de execução das instruções existente entre elas. Se executarmos:

```
tic;
comando1;
comando2;
intervalo = toc;
```

   Então a variável `intervalo` conterá o tempo (em segundos) que o MatLab levou para executar os comandos `comando1` e `comando2`.

   Com base nisso, construa um programa que execute a função `ordena_bolha()` para vetores aleatórios (criados usando a função `randn()` do MatLab) de tamanho $20, 40, 60, ..., 2000$. Armazene o intervalo de execução da ordenação desses vetores aleatórios e monte um gráfico relacionando tamanho versus intervalo de execução.

3. Dado um vetor qualquer, crie uma função que retorne as posições dos três maiores valores presentes nele. O protótipo da função será:

```
[posicao] = acha_3_superiores(vetor)
```

4. Generalize `acha_3_superiores()` para retornar as posições dos k menores valores presentes nele, ou seja:

```
[posicao] = acha_inferiores(vetor, k)
```

5. Crie uma função que retorne a posição dos pontos de máximo (derivada primeira do sinal igual a zero) presentes em um vetor qualquer, usando o seguinte protótipo:

```
[posicao] = acha_maximos(vetor)
```

Considere que um ponto de máximo na posição k de um vetor v é definida por:

$$v[k-1] \leq v[k] \geq v[k+1] \tag{1.1}$$

6. Dado três pontos equidistantes, crie uma função que retorne a posição do valor máximo se tais pontos forem interpolados parabólicamente. O protótipo da função é:

```
[posicao_maximo] = interpola_parabola(v1, v2, v3)
```

Capítulo 2

Problemas genéricos

1. Crie o seguinte vetor para ser usado nos problemas abaixo:

   ```
   entrada = randn(10000, 1)
   ```

   Você usará este vetor para facilitar as análises dos resultados dos programas a seguir:

2. Crie um programa que some todos os elementos de um vetor, retornando tal resultado. O nome do arquivo será `soma.m` e deve usar a seguinte sintaxe:

   ```
   resultado = soma(vetor);
   ```

3. Crie um programa que calcule a média e o desvio-padrão de um vetor, retornando tais resultados. Sua sintaxe será:

   ```
   [media, desvio_padrao] = estatistica(vetor);
   ```

4. Crie um programa que a partir de um vetor fornecido, calcule o módulo de cada elemento desse vetor, retornado o novo vetor. Sua sintaxe será:

   ```
   [vetor_resultado] = modulo_vetor(vetor);
   ```

5. Crie um programa que calcule a média da diferença finita de primeira ordem de um vetor qualquer, ou seja:

   $$\text{vetor}[i] - \text{vetor}[i-1] \tag{2.1}$$

6. Faça um programa que navegue sobre um vetor qualquer, de n em n passos e mostre w amostras do sinal. O deslocamento n e o comprimento da janela w devem ser especificados pelo usuário (usando o comando `input`, do MATLAB) e a cada passo, o MATLAB deve esperar que o usuário pressione uma tecla qualquer (comando `pause`).

   Use o comando `plot()` para mostrar a janela. Use os comandos `sprintf` e `title` para formatar e apresentar no título do desenho, respectivamente, usando a expressão:

   ```
   <n>-ÉSIMA JANELA DE COMPRIMENTO <w>
   ```

   Onde `<n>` e `<w>` devem ser substituídos pelos deslocamento e comprimento usados para desenhar aquele trecho do vetor.

Capítulo 3

Energia e ZCR

1. Use o comando `wavread` para ler arquivos no formato `.wav`. Tal comando pode retornar, além do vetor contendo as amostras do sinal contida no arquivo lido, a taxa de amostragem (em Hz) e a taxa de quantização (em bits por amostra). Use o comando `help` para entender melhor os parâmetros de entrada e de retorno da função.

   Caso o arquivo contenha sinal estéreo, o comando retornará uma matriz de duas colunas, cada uma contendo um dos canais (direito e esquerdo) do sinal estéreo. Use `size` para confirmar as dimensões da matriz retornada.

2. Crie um programa que aceite como entrada:

   - Vetor de dados
   - Tamanho da janela
   - Deslocamento entre janelas

   e calcule, janela a janela, a estimativa de energia do sinal. A energia é calculada em um vetor de comprimento M por:

   $$\text{Energia(vetor)} = \frac{\sum_{i=1}^{M} |\text{vetor}[i]|^2}{N} \tag{3.1}$$

   Como resultado, o programa deve retornar um vetor cujo conteúdo corresponde à estimativa local de energia do sinal.

3. Crie um programa que aceite como entrada:

   - Vetor de dados
   - Tamanho da janela
   - Deslocamento entre janelas

   e calcule, janela a janela, a média de cruzamentos de zeros. Um cruzamento por zero é contado quando:

   $$\text{ZCR}_n = \begin{cases} 1, & x[n-1] < 0 \wedge x[n] > 0 \\ 1, & x[n-1] > 0 \wedge x[n] < 0 \\ 0, & \text{c.c.} \end{cases} \tag{3.2}$$

   ou,

$$\mathrm{ZCR}_n = \begin{cases} 1, & x[n-1]x[n] < 0 \\ 0, & \text{c.c.} \end{cases} \tag{3.3}$$

O problema reside no fato de como lidar com $x[n] = 0$. Isso fica a seu critério neste exercício. A taxa de cruzamentos de zeros é:

$$\mathrm{ZCR} = \frac{\sum_{i=1}^{N} \mathrm{ZCR}_i}{N} \tag{3.4}$$

Como resultado, o programa deve retornar um vetor cujo conteúdo corresponde à taxa de cruzamento por janela.

Capítulo 4

Eliminação do nível DC

Note que muitas vezes, em regiões da locução onde não há qualquer fonema pronunciado (nitidamente uma região contendo apenas o ruído ambiente), o sinal contém um nível DC que afeta diretamente a quantificação da ZCR.

Para eliminá-la, pode-se calcular e eliminar a média do sinal no trecho (eliminado-se apenas o nível DC) ou estimar a curva de tendência das amostras (uma linha reta, para simplificar, via método dos mínimos quadrados) e removê-la. Tais processos podem ser simplicados se aplicarmos um filtro passa-alta simples.

Tal filtro e suas variantes podem representar os efeitos de irradiação na boca durante a fonação.

1. Aplique o filtro

$$R(z) = 1 - 0.95z^{-1} \tag{4.1}$$

   e recalcule a ZCR do sinal. Mostre ambos os sinais (antes e depois da filtragem) na mesma janela de resultados para facilitar a comparação visual.

2. Aplique o filtro

$$R(z) = 1 - 0.99z^{-1} \tag{4.2}$$

   e recalcule a ZCR do sinal. Mostre ambos os sinais (antes e depois da filtragem) na mesma janela de resultados para facilitar a comparação visual.

3. Mostre as ZCRs dos sinais para ambos os filtros e avalie das diferenças.

4. Plote a resposta em frequência de ambos os filtros (4.1 e 4.2) usando o comando `freqz()`. Não esqueça de usar a frequência de amostragem em um dos parâmetros desse comando para que os resultados indiquem qual a frequência de corte facilmente.

5. Plote o diagrama de pólos e zeros de ambos os filtros (4.1 e 4.2) usando o comando `zplane()`.

Capítulo 5

Atraso temporal para energia e ZCR

O tamanho da janela influencia as medidas de energia e de taxa de cruzamento por zeros ao longo do sinal de fala. Para melhorar a estimativa, ao invés de alterar o seu tamanho, iremos deslocá-la no tempo e verificar como tais medidas são influenciadas.

1. Calcule a energia e atraso sem qualquer atraso temporal.
2. Recalcule a energia e atraso para um atraso temporal de $\frac{1}{8}$ do tamanho da janela.
3. Recalcule a energia e atraso para um atraso temporal de $\frac{1}{4}$ do tamanho da janela.
4. Recalcule a energia e atraso para um atraso temporal de $\frac{1}{2}$ do tamanho da janela.
5. Recalcule a energia e atraso para um atraso temporal de $\frac{3}{4}$ do tamanho da janela.
6. Plote as curvas de energia simultaneamente e compare os resultados, particularmente nas regiões onde a derivada da energia é elevada (regiões de transição). Para essas regiões, plote o sinal de voz e verifique qual atraso representou melhor a transição.
7. Plote as curvas de ZCR simultaneamente e compare os resultados, particularmente nas regiões onde a derivada da energia é elevada (regiões de transição). Para essas regiões, plote o sinal de voz e verifique qual atraso representou melhor a transição.

Capítulo 6

Autocorrelação e Correlação cruzada

Uma de suas funções é avaliar o grau de similaridade entre dois sinais distintos (correlação cruzada) ou trechos distintos do mesmos sinal (autocorrelação).

A definição da correlação cruzada entre dois sinais discretos ($x[n]$ e $y[n]$) é:

$$R_{xy}[k] = R^*_{xy}[-k] = \sum_{n=-\infty}^{\infty} x^*[n]\ y[n+k] \tag{6.1}$$

A definição da autocorrelação de um sinal discreto ($x[n]$) é:

$$R_{xx}[k] = R^*_{xx}[-k] = \sum_{n=-\infty}^{\infty} x^*[n]\ x[n+k] \tag{6.2}$$

Assumindo ergodicidade das seqüências e considerando que o tamanho das seqüências são limitadas (ambas de 1 até N), temos estimativas ($\widehat{R}$) da correlação cruzada e autocorrelação, respectivamente:

$$\widehat{R}_{xy}[k] = \begin{cases} A \sum_{n=1}^{N-k} x^*[n]\ y[n+k] & k \geq 0 \\ \widehat{R}^*_{xy}[-k] & k < 0 \end{cases} \tag{6.3}$$

$$\widehat{R}_{xx}[k] = \begin{cases} A \sum_{n=1}^{N-k} x^*[n]\ x[n+k] & k \geq 0 \\ \widehat{R}^*_{xx}[-k] & k < 0 \end{cases} \tag{6.4}$$

Com $-N < k < N$.

Note que A é uma constante e que é definida de acordo com o tipo de estimação (propriedades estatísticas) se deseja. Numericamente reflete um fator de escala para compensar o número limitado de amostras na computação das correlações.

Casos típicos:

- $A = 1$

  Segue a definição matemática mas implica em $\widehat{R}_{xy}[k]$ ou $\widehat{R}_{xx}[k] \to \infty$ com o aumento de N.

- $A = \frac{1}{N}$

  Normaliza a estimativa mas torna-a tendenciosa ('*biased*').

- $A = \frac{1}{N-|k|}$

  Normaliza a estimativa mas torna-a não-tendenciosa (‘*unbiased*’).

1. Desenvolva uma função `autocorr(x, tipo)`

```
vetor = [1 1 1];
resultado1 = autocorr(vetor, 'normal');
resultado2 = autocorr(vetor, 'biased');
resultado3 = autocorr(vetor, 'unbiased');
```

2. Compare os valores produzidos por seu algoritmo com aqueles gerados pelo Matlab (usando a função `xcorr()`). Verifique seus parâmetros para saber como definir A.

3. Faça um gráfico comparativo de velocidade entre seu programa e a versão implementada pelo Matlab variando o comprimento do sinal de 100 até 10000. Use as funções `tic` e `toc` para controlar o tempo gasto para processamento.

```
sinal = randn(1, 1000);
identificador = tic;
xcorr(sinal)
tempo_gasto = toc(identificador);

identificador = tic;
autocorr(sinal)
tempo_gasto = toc(identificador);
```

4. Calcule a autocorrelação de um sinal senoidal e plote seu resultado para cada tipo de autocorrelação (pode sobrepor os resultados). Para gerar o sinal senoidal use:

```
tempo = 0:0.01:5;
senoide = sin(2*pi*tempo);
```

5. Calcule a correlação cruzada entre os seguintes sinais e plote seu resultado para cada tipo de correlação cruzada (pode sobrepor os resultados):

```
tempo = 0:0.01:5;
senoide = sin(2*pi*tempo);
triangular = sawtooth(5*pi*tempo);
```

6. De modo similar ao programa criado para calcular a energia e ZCR, janela-a-janela, e usando a função `xcorr()` do Matlab, determine o vetor autocorrelação, janela-a-janela de um sinal qualquer, tendo como entrada

   - Vetor de dados
   - Tamanho da janela
   - Deslocamento entre janelas

   Nos cálculos de energia e ZCR, obtinhamos um vetor de valores (de energia e ZCR). Como `xcorr()` retorna um vetor, o resultado final será agora um vetor de vetores (de autocorrelação), ou seja, uma matriz. Plote esta matriz usando o operador `mesh()` e/ou `surf()`.

Capítulo 7

Transformada discreta de Fourier

A DFT (Transformada discreta de Fourier) é calculada através do operador `fft()`, e sua inversa através do operador `ifft()`. Basicamente aplicada a sinais (pode ser aplicada a matriz, que é interpretada como um conjunto de vetores).

1. Calcule a DFT de um sinal senoidal contendo três ciclos.

```
t_amostragem = 0.01;
tempo = 0:t_amostragem:3;
senoide = sin(2*pi*tempo);
senoide_dft = fft(senoide)/length(senoide);
frequencia = (0:length(senoide_dft)-1)/length(senoide_dft)/t_amostragem;
subplot(2, 2, 1);
stem(frequencia, abs(senoide_dft));
subplot(2, 2, 2);
stem(frequencia, angle(senoide_dft));
subplot(2, 2, 3);
stem(frequencia, abs(senoide_dft));
axis([-1 5 0 1]);
subplot(2, 2, 4);
stem(frequencia, angle(senoide_dft));
axis([-1 5 -4 +4]);
```

2. Assumindo que o sinal senoidal (de três ciclos) tenha tamanho N, recalcule a DFT de $10N$ pontos, incluindo zeros ao final do sinal.

```
t_amostragem = 0.01;
tempo = 0:t_amostragem:3;
senoide = sin(2*pi*tempo);
tamanho_dft = 10*length(senoide);
senoide_dft = fft(senoide, tamanho_dft)/length(senoide);
frequencia = (0:tamanho_dft-1)/tamanho_dft/t_amostragem;
subplot(2, 2, 1);
stem(frequencia, abs(senoide_dft));
axis([frequencia(1) frequencia(length(frequencia)) 0 1]);
subplot(2, 2, 2);
stem(frequencia, angle(senoide_dft));
axis([frequencia(1) frequencia(length(frequencia)) -4 +4]);
```

```
subplot(2, 2, 3);
stem(frequencia, abs(senoide_dft));
axis([-1 5 0 1]);
subplot(2, 2, 4);
stem(frequencia, angle(senoide_dft));
axis([-1 5 -4 +4]);
```

3. Que evidência você consegue extrair do processo de inclusão de zeros (via parâmetro adicional em `fft()`) no cálculo da DFT de um sinal qualquer?

4. Que evidência de simetria você enxerga na magnitude e fase do sinal? Se tais simetrias ocorrem, qual a justificativa para sua existência?

5. Construa uma onda quadrada periódica (usando `square()`) e calcule a DFT usando exatos 10 ciclos da onda (sem acrescentar zeros ao sinal).

   Use como frequência de amostragem $f_a = 10 \times (1/T)$, sendo T o período mínimo dessa onda. Como resultado, mostre (`stem()` ou `plot()`) a magnitude (módulo) e a fase da DFT apenas para as componentes significativas (As componentes significantes são aquelas com elevada magnitude - use o zoom horizontal para identificar tais componentes).

6. Repita o procedimento anterior para uma onda triangular (usando `sawtooth()`).

7. Repita os dois procedimentos anteriores, calculando agora a magnitude em dB (ou seja, $20 \log_{10}(\text{abs}(\text{dft}))$).

8. Crie um programa que aceite como entrada:

   - Vetor de dados
   - Tamanho da janela
   - Tamanho da DFT ($\geq$ tamanho da janela)
   - Deslocamento entre janelas

   e calcule, janela a janela, a magnitude espectral (em dB) do sinal. Como a cada janela obteremos um vetor, o resultado será um vetor de vetores (ou seja, uma matriz). Plote esta matriz usando o operador `mesh()` e/ou `surf()`. O resultado será o espectrograma do sinal.

   Use `axis()` e `view()` para ajustar as magnitudes de visualização. Use `colormap()`, `shading()` e `hidden` para alterar o padrão gráfico do resultado.

   Compare o resultado com o operator `specgram()`.

9. Repita o procedimento anterior, agora calculando, janela a janela, a fase do sinal.

10. Crie um programa que aceite como entrada:

    - Vetor de dados
    - Tamanho da janela
    - Tamanho da DFT ($\geq$ tamanho da janela)
    - Deslocamento entre janelas

    e calcule, janela a janela, a magnitude espectral (em dB, a energia e a ZCR do sinal. Mostre os resultados na mesma janela gráfica, usando `subplot()` para que os três resultados sejam visualizados simultaneamente.

11. Considere o seguinte sinal:

$$x(t) = 5\cos(2000\pi t + \pi/4) + 3\cos(2200\pi t) - 5\,\mathrm{sen}(9000\pi t) \tag{7.1}$$

Defina a frequência de amostragem adequada para discretizar esse sinal. Aplicando a DFT sobre o sinal discretizado, identifique as magnitudes e fases das senóides e cossenóides que compõem o sinal.

Lembre-se que você deve considerar uma quantidade mínima de amostras igual a 2 ou 3 ciclos do período mínimo do sinal. Também considere o uso de zeros ao final da seqüência (via operando de `fft()`) para tentar melhorar a resolução espectral dos picos de magnitude.

Uma vez encontrada a quantidade mínima de amostras (mais de 2 ciclos) e tamanho da DFT (via zeros ao final da seqüência), plote a magnitude e fase na região ao redor das três componentes espectrais de $x(t)$.

12. Repita o procedimento anterior, adicionando ruido gaussiano à seqüência discretizada do sinal:

$$x(t) = 5\cos(2000\pi t + \pi/4) + 3\cos(2200\pi t) - 5\,\mathrm{sen}(9000\pi t) \tag{7.2}$$

Supondo que `x[n]` é seu sinal discretizado, analise `x_ruido[n]` (que é definido por):

```
x = ... % discretização de x(t)
x_ruido = x + 2 * randn(size(x));
```

Capítulo 8

Convolução e Filtragem

A filtragem envolve uma convolução entre um sinal qualquer e a resposta ao impulso do filtro, se considerarmos que o sistema que representa o filtro é linear e invariante no tempo. Caso contrário, devemos resolver (analiticamente ou numericamente) a equação (geralmente diferencial) que representa o sistema.

A convolução naturalmente é definida por:

$$y[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] = \sum_{k=-\infty}^{+\infty} x[n-k]h[k] \tag{8.1}$$

1. Selecione um sinal de voz qualquer (que chamaremos de $x[n]$). Crie um programa que realize uma filtragem do tipo média móvel com duração N a ser definida como parâmetro de entrada. Plote tanto o sinal de entrada ($x[n]$ quanto o sinal resultante da filtragem.

   O filtro média móvel em consideração possui resposta ao impulso definida por:

$$h[n] = \frac{1}{N} \sum_{k=-N}^{+N} \delta[n] \tag{8.2}$$

2. Visualize a resposta em frequência do filtro (que sabidamente é fase zero) (usando o comando `zerophase()` informando apropriadamente a frequência de amostragem para correta interpretação dos resultados). Note que tal filtro média móvel possui apenas coeficientes b, sendo $a = 1$.

3. Considere a seguinte equação a diferenças finitas, linear, com coeficientes constantes:

$$\sum_{i=0}^{N-1} a_i y[n-i] = \sum_{j=0}^{M-1} b_j x[n-j] \tag{8.3}$$

   Crie um programa de filtragem (`meu_filtro(a, b, x)`) que aceite como entrada:

   - Vetor com os N coeficientes a_i
   - Vetor com os M coeficientes b_j
   - Vetor com a sequência a ser processada

e calcule o resultado da EDO com condições iniciais para $y[n]$ e $x[n]$ nulas para $n < 0$. O resultado será uma sequência de dados com o mesmo tamanho de $x[n]$. Para testes, use:

$$\begin{aligned} a &= [1, 0.7, 0.1]' \\ b &= [1, 1.6, 1.64]' \\ x &= [1, 2, 3, 1, 2, 3, 1, 2, 3]' \end{aligned} \tag{8.4}$$

Que produzirá

$$y = [1, 2.9, 5.71, 4.793, 4.5939, 4.145, 5.7191, 4.1021, 4.3966]' \tag{8.5}$$

Compare o resultado do seu programa com a função `filter()` do MatLab.

Capítulo 9

Projeto de filtros

1. Faça um programa que apresente magnitude (em dB, usando a `fft()`) das seguintes janelas espectrais: Retangular, Triangular, Hanning, Hamming e Blackman. Todas as curvas devem ser apresentadas em uma única representação gráfica (usando cores distintas e legenda para facilitar a compreensão). Considere que o intervalo de frequência é normalizado (entre 0 e pirad/sec).

2. Usando os comandos `buttord()`, `butter()`, projete um filtro analógico (domínio s) do tipo passa alta com os seguintes requisitos:

$$\begin{array}{l} -20\text{dB}, f < 700\text{Hz} \\ -2\text{dB}, f > 1200\text{Hz} \end{array} \tag{9.1}$$

   Visualize a resposta em frequência desse filtro. Pode usar o comando `bode()` (e provavelmente `tf()`, `zpk()` ou `ss()`) antes da geração da resposta em frequência analógica) ou o comando `freqs()`. Mostre visualmente que os requisitos foram atingidos.

3. Assumindo uma frequência de amostragem de ($fs =$) 4800Hz, converta o filtro analógico anterior para seu equivalente digital, usando o comando `bilinear()`. Mostre visualmente que os requisitos foram atingidos.

   Visualize sua resposta em frequência usando o comando `freqz()` e seu diagrama de pólos e zeros (`zplane()` e `zp2tf()`).