PROCESSAMENTO DIGITAL DE SINAIS

SLITD’s e Convolução

# Fundamentação Teórica

Matematicamente, um sistema de tempo discreto (*sistema discreto*) é descrito por um operador *T*{⋅} que transforma uma sequência de valores de entrada ***x*[*n*]** (*sinal excitação*) em uma sequência de valores de saída ***y*[*n*]** (*sinal resposta*):

*y*[*n*] *= T*{ *x*[*n*] }(1)

Sistema Discreto

Linear

Em PDS, dizemos que o sistema processa o sinal de *entrada* gerando o sinal de *saída*. Sistemas discretos podem ser classificados em sistemas *lineares* e sistemas *não lineares*. Um sistema discreto *T*{⋅} será um operador linear se *T*{⋅} satisfizer o **princípio da superposição**:

*T*{ *a1.x1*[*n*] *+ a2.x2*[*n*] } = *a1.T*{ *x1*[*n*] } + *a2.T*{ *x2*[*n*] }, ∀ *a1, a2, x1*[*n*]*, x2*[*n*](2)

Decomposição em Impulsos Unitários

Qualquer sequência arbitrária *x*[*n*]pode ser decomposta numa soma de impulsos unitários escalados em amplitude e deslocados no tempo:



(3)

Usando as equações (2) e (3), a saída ***y*[*n*]** de um sistema linear gerada a partir de uma entrada arbitrária ***x*[*n*]** será dada por:



(4)

A resposta *T*{*δ*[*n - k*] } pode ser interpretada como sendo a resposta de um sistema linear no tempo *n* devido a um impulso unitário ocorrido no tempo *k*. Ela é chamada de ***resposta impulsiva*** e é denotada por ***hk*[*n*]**. A saída completa é dada pela *soma da superposição*



(5)

O cálculo da equação (5) requer uma resposta impulsiva variante no tempo *hk*[*n*], a qual na prática é muito difícil de ser obtida. Por isso, sistemas de tempo invariante serão considerados.

Sistema

Linear

Invariante no Tempo

Um sistema linear no qual o par de sinais entrada-saída, ***x*[*n*]** e ***y*[*n*]**, são invariantes a deslocamentos *k* no tempo é chamado de *Sistema Linear Invariante no Tempo Discreto* (*SLITD*). Para um SLITD os operadores: *T*{⋅} e deslocamento no tempo são intercambiáveis, como mostrado abaixo:

*x*[*n*]

*y*[*n*]

*T*{⋅}

Desloc. por *k*

*y*[*n-k*]

*x*[*n*]

*T*{⋅}

Desloc. por *k*

*x*[*n-k*]

*y*[*n-k*]

A resposta impulsiva variante no tempo, ***hk*[*n*]**, torna-se uma função invariante no tempo para um SLITD: *h*[*n ‑ k*], e portanto, a saída *y*[*n*] do SLITD será dada por:

Convolução



(6)

A resposta impulsiva de um SLIT é dada por *h*[*n*]. A operação matemática definida pela equação (6) é chamada de ***soma da convolução linear*** e é denotada por



(7)

Portanto, pode-se dizer que um SLIT é completamente caracterizado no domínio do tempo por sua resposta impulsiva *h*[*n*]:

*x*[*n*]

*y*[*n*]*=x*[*n*]*\*h*[*n*]

*h*[*n*]

A operação de convolução de dois sinais pode ser calculada de vários modos. Se as sequências são funções matemáticas (de duração finita ou infinita), pode-se avaliar analiticamente a equação (6) para todo *n*, obtendo-se a forma analítico-funcional de *y*[*n*]. Mas se as sequências são sinais representados por amostras, então a convolução pode ser realizada via uso de processadores digitais de sinais (DSP’s) que usam recursos computacionais para fazer os cálculos em tempo real.

# Exemplos

1. **Componentes simétricas** dos sinais: todo sinal pode ser decomposto em suas componentes de simetria par e de simetria ímpar:



Use a função **[xe,xo,m] = evenodd(x,n)** para decompor em componentes simétricas o sinal *boxcar* deslocado, *x*[*n*] = *u*[*n*] – *u*[*n* ‑3].

Verifique graficamente o resultado da decomposição e comprove a recuperação do sinal original a partir da soma das componentes.

**Python**

**# -\*- coding: utf-8 -\*-**

**""" Decomposição de sinal em componentes simétricas: par e ímpar**

**@author: Prof. Cláudio, Ago/2014 """**

**from numpy import arange, zeros, imag**

**from matplotlib.pyplot import subplot, stem, title, xlabel, ylabel, axis, grid**

**def stepseq(n0=0, ne=-10, nd=10):**

**# Geração da sequência degrau unitário deslocada de 'n0'**

step = degrau

seq = sequência

*gera um sinal degrau unitário com desloc. ‘n0’*

**# amostras à direita, no intervalo de tempo de 'ne' a 'nd'**

**# --------------------------------------------------------**

**# u, n = stepseq(n0, ne, nd)**

**n = arange(ne,nd+1) # base temporal**

**u = zeros(len(n))**

**u[n >= n0] = 1. # amplitude do sinal degrau**

**return u, n**

**def evenodd(x,n):**

**# Decomposição de sinal real em componentes par e ímpar**

**# -----------------------------------------------------**

even = par

odd = ímpar

*calcula as componentes*

*par e ímpar do sinal ‘x’*

**# xe, xo, m = evenodd(x,n)**

**if any(imag(x) != 0):**

**print u'sinal x não é uma sequência real!'**

**return 0,0,0**

**m = -n[::-1]**

**m1 = min([min(m),min(n)])**

**m2 = max([max(m),max(n)])**

**m = arange(m1,m2+1) # nova base temporal**

**n1 = arange(0,len(n))**

**x1 = zeros(len(m))**

**nm = n[0]-m[0] # distância entre as bases temporais em unid.s de tempo**

**x1[n1+nm] = x**

**xe = 0.5\*(x1 + x1[::-1])**

**xo = 0.5\*(x1 - x1[::-1])**

**return xe, xo, m**

**# criando sinal: pulso digital (sinal porta)**

cria um sinal porta, a ser

decomposto em compon.s par e ímpar...

**u1, n = stepseq(0,-5,5)**

**u2, n = stepseq(3,-5,5)**

**x = u1 - u2**

**[xe,xo,m] = evenodd(x,n) # decomposição par e ímpar**

**xr = xe + xo; # sinal recuperado**

**# gráficos**

**subplot(2,2,1); stem(n,x); grid('on'); axis([-5, 5, -0.2, 1.2])**

**title(r'Sinal Original: $\itx\rm[\itn\rm]$')**

**xlabel(r'$\itn$'), ylabel(r'$\itx\rm[\itn\rm]$')**

**subplot(2,2,2); stem(m,xr); grid('on'); axis([-5, 5, -0.2, 1.2])**

**title(r'Sinal Recuperado: $\itx\_e\rm[\itn\rm] + \itx\_o\rm[\itn\rm]$')**

**xlabel(r'$\itn$'), ylabel(r'$\itx\_r\rm[\itn\rm]$')**

**subplot(2,2,3); stem(m,xe); grid('on'); axis([-5, 5, -1.2, 1.2])**

**title('Componente PAR')**

**xlabel(r'$\itn$'); ylabel(r'$\itx\_e\rm[\it\rmn]$')**

**subplot(2,2,4); stem(m,xo); grid('on'); axis([-5, 5, -1.2, 1.2])**

**title(u'Componente ÍMPAR')**

**xlabel(r'$\itn$'); ylabel(r'$\itx\_o\rm[\itn\rm]$')**

1. **Convolução linear**, dada pelas equações (6) e (7).

Use a função **y = convolve(x,h)** do pacote **scipy.signal** para convoluir duas sequências discretas de duração finita. A função *'convolve'* assume que as duas sequências iniciam-se na origem dos tempos, *n* = 0. Digite o *script* seguinte para convoluir e mostrar os gráficos das sequências  *x*[*n*] = {3, 2, 1, 0, -2, 4, 2} e *h*[*n*] = {3, 3, 0, -5, 1, 2}.

**# -\*- coding: utf-8 -\*-**

**""" Convolução de sinais discretos**

**@author: Prof. Cláudio, Ago/2014 """**

**from** numpy **import** array,arange

**from** scipy.signal **import** convolve

**from** matplotlib.pylab **import** subplot, stem, title, xlabel, ylabel, axis, grid

**# criando sinal: pulso digital (sinal porta)**

**x = array([3, 2, 1, 0, -2, -1, 2]); nx = arange(0,len(x))**

**h = array([3, 3, 0, -2, 1, 2]); nh = arange(0,len(h))**

**y = convolve(x,h); ny = arange(0,len(y))**

**# gráficos**

**subplot(2,2,1), stem(nx,x,linewidth=3); axis([-1,12,-3,5])**

**title(r'Sinal de Entrada: $\itx\rm[\itn\rm]$')**

**xlabel(r'$\itn$'); ylabel(r'$\itx\rm[\itn\rm]$'); grid('on')**

**subplot(2,2,2), stem(nh,h); axis([-1,12,-3,4])**

**title(r'Resp. ao Impulso: $\ith\rm[\itn\rm]$'),**

**xlabel(r'$\itn$'); ylabel(r'$\ith\rm[\itn\rm]$'); grid('on')**

**subplot(2,2,3), stem(ny,y); axis([-1,12,-10,16])**

**title(u'Sinal de Saída: ' + r'$\ity\rm[\itn\rm]$')**

**xlabel(r'$\itn$'); ylabel(r'$\ity\rm[\itn\rm]$'); grid('on')**

Use a função modificada, **[y,ny] = conv\_m(x,nx,h,nh)** para convoluir sequências que se iniciam em qualquer instante. Digite o *script* seguinte para convoluir e mostrar os gráficos das sequências *x*[*n*] = {3, 2, 1, 0, -2, 4, 2} e *h*[*n*] = {3, 3, 0, -5, 1, 2}.

**# -\*- coding: utf-8 -\*-**

**""" Convolução de sinais discretos, Prof. Cláudio, Ago/2014 """**

**from numpy import array, arange**

**from scipy.signal import convolve**

**from matplotlib.pyplot import subplot, stem, title, xlabel, ylabel, axis, grid**

**def convolve\_m(x,nx,h,nh):**

**""" Convolução modificada: retorna x \* h e a sua referência temporal**

----------------------------------------------------------------

y,ny = convolve\_m(x,nx,h,nh)

y,ny - resultado da convolução (ordenadas e abscissas)

x,nx - sinal de entrada (ordenadas e abscissas)

h,nh - sinal resposta ao impulso unitário (ordenadas e abscissas) **"""**

**linf = nx[0] + nh[0]**

**lsup = nx[-1] + nh[-1]**

**ny = arange(linf,lsup+1)**

**y = convolve(x,h)**

**return y, ny**

**# criando sinal: pulso digital (sinal porta)**

**x = array([3, 2, 1, 0, -2, -1, 2]); nx = arange(-3,4)**

**h = array([3, 3, 0, -2, 1, 2]); nh = arange(-1,5)**

**y, ny = convolve\_m(x,nx,h,nh)**

**# continua na página seguinte --->**

**# gráficos**

**subplot(2,2,1), stem(nx,x,linewidth=3); axis([min(nx)-1,max(nx)+1,-3,5])**

**title(r'Sinal de Entrada: $\itx\rm[\itn\rm]$')**

**xlabel(r'$\itn$'); ylabel(r'$\itx\rm[\itn\rm]$'); grid('on')**

**subplot(2,2,2), stem(nh,h); axis([min(nh)-1,max(nh)+1,-3,4])**

**title(r'Resp. ao Impulso: $\ith\rm[\itn\rm]$'),**

**xlabel(r'$\itn$'); ylabel(r'$\ith\rm[\itn\rm]$'); grid('on')**

**subplot(2,2,3), stem(ny,y); axis([min(ny)-1,max(ny)+1,-10,16])**

**title(u'Sinal de Saída: ' + r'$\ity\rm[\itn\rm]$')**

**xlabel(r'$\itn$'); ylabel(r'$\ity\rm[\itn\rm]$'); grid('on')**

# Procedimentos

1. Seja um trem de pulsos, *x*[*n*] *= u*[*n+2*] *– u*[*n*–10], usado como sinal de entrada em um SLITD descrito por sua **resposta impulsiva** *h*[*n*] *= 0.*8n (u[*n*] – *u*[*n*–8]). Determine a saída *y*[*n*] do SLITD, e trace todos os sinais envolvidos.
2. Calcule manualmente a convolução das sequências *x*[*n*] e *h*[*n*], e em seguida verifique computacionalmente o resultado obtido por você, traçando os sinais (entrada, resposta impulsiva e saída) em um único plano cartesiano.

*x*[*n*] *=* [11, 7, 0, -1, 4] e *h*[*n*] *=* [ 3, 0, -5, 2, 1],

1. Monte o mesmo sistema da aula anterior (PC, Python, placa AQ.-USB, Ger. de Sinais, Osc.), para amostragem e quantização de sinais:



1. Utilizando o *script* Python de apresentação de forma de onda de sinal com interface gráfica em tempo real, desenvolvido também na aula anterior, acrescente o processamento relativo à aplicação do sistema proposto no item anterior, exibindo o sinal de entrada e o de saída na mesma tela (dois traços de cores diferentes com legenda de identificação). *Dica*: represente o SLIT através de sua EDLCC, convertendo a Resposta ao Impulso.

**☝ Atenção**: Cuidado para não **curto-circuitar** a entrada da placa de aquisição durante a montagem do circuito, pois sérios danos podem resultar dessa operação, a você e ao equipamento.

# Exercícios

1. Crie um *script* para esboçar os seguintes sinais em uma mesma figura, com -10 ≤ *n* ≤ 10: 5δ[n], ‑5*u*[*n*], 5*u*[*n*-10], 0,5*nu*[*n*], 3sen(0.2π*n*), 0,5*n*sen(π.*n*/10)*u*[*n*].
2. Plote os sinais *x*[*n*] = *n.u*[*n*-6], *y*[*n*] = *n.u*[6-n] e a convolução deles, em um mesmo plano cartesiano, com -10 ≤ *n* ≤ 10.
3. Supondo um DSP com intervalo de amostragem de *Ts* = 5,0 ms, converta para digital cada um dos seguintes sinais analógicos, quantizando-os com 4 bits na faixa de -1,0 V a 1,0 V, e depois calcule e trace a convolução das sequências para 0,0 < t < 1,0 s:
   1. *h*(*t*) = *e*-5*t u*(*t*) b. *x*(*t*) = 0,5*.*sen(20*πt*) *u*(*t*)

**from** numpy **import** arange,sin,exp,pi,linspace,zeros

**from** pylab **import** plot, stem, subplot

**from** scipy.signal **import** convolve

**def** quantize(sd,bits,fd):

''' sd - sinal discreto (amplitude com valor de precisão infinita)

bits - 2^bits = quantidade de níveis de quantização

fd - faixa dinâmica: valor mínimo e valor máximo '''

val = linspace(fd[0],fd[1],2\*\*bits)

sq = zeros(len(sd)) # sinal quantizado

i = 0

**for** amp **in** sd:

sq[i] = val[abs(sd[i]-val).argmin()]

i += 1

**return** sq

Ts = 5e-3 # intervalo de tempo entre amostras (s)

fs = 1./Ts # amostras por segundo

t = arange(0,1+Ts,Ts)

x = 0.5\*sin(2\*pi\*10\*t) # sinal de entrada no SLITD

xq = quantize(x,4,[-1.,1.]) # entrada quantizada

h = exp(-5.\*t) # resposta ao impulso do SLITD

hq = quantize(h,4,[-1.,1.]) # resposta ao impulso quantizada

y = convolve(x,h)

yq = convolve(xq,hq)

subplot(311); plot(t,x,'b'); stem(t,xq,'r')

subplot(312); plot(t,h,'b'); stem(t,hq,'r')

subplot(313); plot(y,'b'); stem(yq,'r')

1. Classifique os seguintes sistemas em relação à linearidade e à invariância no tempo? Justifique suas respostas.
   1. *y*[*n*] = 5*x*[*n*] + 2*x*2[*n*]
   2. *y*[*n*] = *x*[*n* – 1] + 4*x*[*n*]
   3. *y*[*n*] = 4*x*3[*n* – 1] + 2*x*[*n*]
2. Calcule de forma manuscrita a convolução de *x*[*n*] = { -2, -2, -2, 1, 1 } com   
   *z*[*n*] = { -1, 2, ‑1 } usando os métodos intuitivo, gráfico, folha deslizante e definição.
3. Pesquise sobre convolução circular e discorra sobre as diferenças e aplicações de cada um dos tipos de convolução.
4. Pesquise sobre correlação e autocorrelação, e em seguida desenvolva um *script* para traçar o gráfico da correlação dos sinais do exercício 2. Você consegue perceber alguma relação dessa figura estatística com a operação convolução? Justifique.

# Apêndice

**String's Modifiers** (modificadores de texto): a contra barra (\) precede todas as sequências de caracteres *TEX*.

|  |  |
| --- | --- |
| Sequência de caracteres *TEX* | Efeito |
| **\bf** | *bold face* (negrito) |
| **\it** | *italics* (itálico) |
| **\rm** | *restore normal font* (volta à fonte normal) |
| **\_{***índice***}** | sub-índice |
| **^{***potência***}** | potenciação |
| **\pi, \mu, \alpha, \beta, \gamma** | Letras grega: π μ α β γ... |
| **\color{***colorname***}** | oito cores básicas: red, green, yellow, magenta, blue, black, White |
| **\color[rgb]{r g b}** | Para especificar uma cor pela soma de cores primárias (RGB) com valores entre 0 e 1 |

**Dica** - editor de equações LaTex online: [www.codecogs.com/latex/eqneditor.php](http://www.codecogs.com/latex/eqneditor.php)

Exemplo:

**# -\*- coding: utf-8 -\*-**

**""" Senóide """**

**import numpy as np**

**import matplotlib.pyplot as plt**

**t = np.arange(0.0, 2.0, 0.01)**

**s = np.sin(2\*np.pi\*t)**

**plt.plot(t,s)**

**plt.title(r'$\alpha\_i > \beta\_i$', fontsize=20)**

**plt.text(1, -0.6, r'$\sum\_{i=0}^\infty x\_i$', fontsize=20)**

**plt.text(0.6, 0.6, r'$\mathcal{A}\mathrm{.sen}(2 \omega t)$', fontsize=20)**

**plt.xlabel('tempo (s)')**

**plt.ylabel(u'tensão (mV)')**

