PROCESSAMENTO DIGITAL DE SINAIS

PRÁTICA

EQUAÇÕES DE DIFERENÇAS & CONVOLUÇÃO

Apresentação

As ***Equações de Diferenças*** são usadas para descrever o comportamento dinâmico de sistemas lineares invariantes no tempo discreto (SLITD). Quando possuem coeficientes constantes, essas equações são denominadas ***Equações de Diferenças Lineares com Coeficientes Constantes*** (EDLCC's).

Neste experimento vamos estudar a definição, as características e formas de solucionar a EDLCC de um SLITD.

Fundamentação Teórica

A forma geral de uma EDLCC para um SLITD com sinais de saída $\boldsymbol{y[n]}$ e de entrada $\boldsymbol{x[n]}$ é:

$$y[n] = \sum_{k=0}^{M} b_k . x[n-k] + \sum_{k=1}^{N} a_k . y[n-k] \tag{1}$$

Outra forma usual para se calcular a saída $\boldsymbol{y[n]}$ de um SLITD a partir da entrada $\boldsymbol{x[n]}$ é a <u>soma de convolução</u>. Um SLITD que possua resposta impulsiva $\boldsymbol{h[n] = \alpha^n . u[n]}$ será descrito pela seguinte EDLCC, gerada a partir da definição da convolução:

$$y[n] = \sum_{k=-\infty}^{\infty} h[k] . x[n-k] = \sum_{k=0}^{\infty} \alpha^k . x[n-k] \tag{2}$$

Do ponto de vista computacional a equação acima não é eficiente, e impraticável por se tratar de uma soma com infinitos termos. Em alguns casos é possível expressar a saída do SLITD em termos de *valores passados da própria saída*, e de valores *corrente e passados da entrada*. O SLITD descrito pela Equação (2) pode ser descrito também pela seguinte EDLCC:

$$y[n] = \alpha . \, y[n-1] + x[n] \tag{3}$$

A Equação (3) calcula a saída $\boldsymbol{y[n]}$ do SLITD para uma entrada arbitrária $\boldsymbol{x[n]}$ com uma única soma, ao invés de infinitas somas, como sugerido pela Equação (2). As equações de diferenças necessitam de um conjunto de ***condições iniciais*** para serem resolvidas. Por exemplo, para uma entrada que se inicia no tempo $\boldsymbol{n = 0}$, a solução da Equação (3) para o mesmo instante de tempo, $\boldsymbol{y[0]}$, pode depender dos valores de $\boldsymbol{y[-1]}$, $\boldsymbol{y[-2]}$, ..., $\boldsymbol{y[-N]}$. Quando as condições iniciais são todas nulas, diz-se que o sistema encontra-se inicialmente em repouso, em estado nulo, ou **relaxado**.

Para um SLITD descrito por uma EDLCC, a resposta impulsiva $h[n]$ pode ser obtida resolvendo-se a Equação (1) para a entrada $x[n] = \delta[n]$ e assumindo o SLITD em repouso.

Supondo um **sistema não recursivo**, $a_k = 0$ para k = 1,2,...N, a Equação (1) torna-se:

$$y[n] = h[n] = \sum_{k=0}^{M} b_k \delta[n-k]$$

Se a resposta impulsiva $h[n]$ for finita, então o **SLITD** é classificado como sendo do tipo **FIR** (***F****inite* ***I****mpulse* ***R****esponse*). Porém, se $a_k \neq 0$, a resposta impulsiva $h[n]$ será de duração infinita, e o **SLITD** será classificado como sendo do tipo **IIR** (***I****nfinite* ***I****mpulse* ***R****esponse*).

Existem vários métodos de resolução de uma EDLCC, para uma determinada entrada $x[n]$ e condições iniciais:

1. **Iterativo**:

   Baseia-se na construção de uma tabela de valores da *entrada* e da *saída*, a partir da avaliação da EDLCC para cada tempo n. Essa abordagem é indicada quando se deseja calcular poucos valores da saída (método essencialmente computacional).

2. **Clássico**:

   Resolução de equações diferenciais (usando autofunções como possíveis soluções da equação), que consiste em obter a solução homogênea[1] e a solução particular[2] (método analítico).

3. **Transformada-Z**:

   Aplicação da Transformada-Z Direta à EDLCC, para obter uma equação algébrica de fácil solução. Em seguida, após a obtenção da solução, aplica-se a Transformada-Z Inversa para obter a resposta no domínio do tempo.

[1] Gerada pelas condições iniciais do sistema.
[2] Gerada pelo estímulo externo (entrada) aplicado ao sistema relaxado.

Procedimentos Práticos

1. Seja um SLITD **relaxado** (c.i.'s nulas) descrito por sua EDLCC de segunda ordem:

$$\boldsymbol{y[n] + 0{,}6\, y[n\text{-}2] = 0{,}3\, x[n] + 0{,}5\, x[n\text{-}1] + 0{,}3\, x[n\text{-}2]}$$

Desenvolva um script para calcular e traçar a resposta do SLITD às seguintes entradas, usando o método iterativo para solucionar a EDLCC:

a) Impulso unitário, $x[n] = \delta[n]$
b) Degrau unitário, $x[n] = u[n]$
c) Exponencial discreta, $x[n] = 3.(0{,}4)^n\, u[n]$

$-5 \leq n \leq 15$

Solução em **Python** *para o item* (a):

```
from numpy import arange, zeros
from pylab import stem, title, xlabel, ylabel, subplot, grid, ylim
n = arange(-5,16)                # base de tempo
N = 2                            # ordem do SLITD: max_atraso(x,y)
imo = 5                          # índice da origem dos tempos
xa = zeros(len(n)); xa[imo] = 1  # entrada a) impulso unitário
ya = zeros(len(n))               # condições iniciais nulas
print ' n    y[n]'
for it in n[N:]:
  m = it + imo
  ya[m] = -0.6*ya[m-2] + 0.3*xa[m] + 0.5*xa[m-1] + 0.3*xa[m-2]   # EDLCC
  print '%2d %+7.5f' % (it,ya[m])
subplot(211); stem(n,xa); title('Entrada - $\delta[n]$')
grid('on'); ylabel('Amplitude'); ylim(-0.5,1.5)
subplot(212); stem(n,ya); title('Resposta ao Impulso - $h[n]$')
grid('on'); xlabel('n'); ylabel('Amplitude')
```

Solução em **Python** *para o item* (b):

```
from numpy import arange, zeros
from pylab import stem, title, xlabel, ylabel, subplot, grid, ylim
n = arange(-5,16)                # base de tempo
N = 2                            # ordem do SLITD: max_atraso(x,y)
imo = 5                          # índice da origem dos tempos
xb = zeros(len(n)); xb[imo:]= 1  # entrada b) degrau unitário
yb = zeros(len(n))               # condições iniciais nulas
print ' n    y[n]'
for it in n[N:]:
  m = it + imo
  yb[m] = -0.6*yb[m-2] + 0.3*xb[m] + 0.5*xb[m-1] + 0.3*xb[m-2]   # EDLCC
  print '%2d %+7.5f' % (it,yb[m])
subplot(211); stem(n,xb); title('Entrada - $u[n]$'); grid('on')
ylabel('Amplitude'); ylim(-0.5,1.5)
subplot(212); stem(n,yb); title('Resposta ao Degrau - $s[n]$')
grid('on'); xlabel('n'); ylabel('Amplitude')
```

Solução em ***Python*** *para o item* (c):

```
from numpy import arange, zeros
from matplotlib.pylab import stem, title, xlabel, ylabel, subplot, grid,
ylim
n = arange(-5,16)                # base de tempo
N = 2                            # ordem do SLITD: max_atraso(x,y)
imo = 5                          # índice da origem dos tempos
xc = zeros(len(n))
xc = 3*(0.4**n) * (n >= 0)       # entrada c) exponencial direita
yc = zeros(len(n))               # condições iniciais nulas
print ' n    y[n]'
for it in n[N:]:
  m = it + imo
  yc[m] = -0.6*yc[m-2] + 0.3*xc[m] + 0.5*xc[m-1] + 0.3*xc[m-2]   # EDLCC
  print '%2d %+7.5f' % (it,yc[m])
subplot(211); stem(n,xc); title(r'Entrada - $3\times(0,4^n)u[n]$');
grid('on'); ylabel('Amplitude'); ylim(-0.5,3.5)
subplot(212); stem(n,yc); title(u'Resposta à Exponencial Direita');
grid('on'); xlabel('n'); ylabel('Amplitude')
```

2. Implemente o sistema do item anterior usando a função de filtragem (`filter` no *Matlab* ou `lfilter` do pacote *scipy.signal* da Ling. *Python*) e calcule novamente as saídas do sistema para as entradas sugeridas.

`y,zf = lfilter(b, a, x, axis=-1, zi=None)`

> *Filtra uma sequência de dados* 'x' *usando um filtro digital definido pelos coeficientes* 'b' *e* 'a'. *Isto vale para muitos tipos de dados fundamentais (incluindo o tipo* ***Object****). O filtro é uma implementação da equação de diferenças padrão na forma transposta II.*

$a[0]*y[n]=b[0]*x[n]+b[1]*x[n-1]+..+b[M-1]*x[n-(M-1)]-a[1]*y[n-1]-...-a[N-1]*y[n-(N-1)]$

Parâmetros:

- *b* *vetor* 1D *com os coeficientes que multiplicam a sequência de entrada*
- *a* *vetor* 1D *com os coeficientes que multiplicam a sequência de saída. Se* '*a*[*0*]' *não for 1, então* '*a*' *e* '*b*' *serão normalizados por* '*a*[*0*]'.
- *x* *vetor N-dimensional com o sinal de entrada.*
- *axis* *valor do tipo* **int** *que indica o eixo dos dados de entrada ao longo do qual será aplicado o filtro linear.*
- *zi* *vetor opcional contendo as condições iniciais para os atrasos do filtro. Ele é um vetor de comprimento* 'max(len(a),len(b))-1'. *Se* '*zi*' *for* **None** *ou não for dado então o sistema será considerado em repouso. Veja* '*lfiltic*' *para mais informações sobre condições iniciais.*

Retorno

- *y* *vetor com a saída do filtro digital*
- *zf* *vetor opcional.*

Se 'zi' for **None***, este vetor não é retornado, caso contrário, 'zf' conterá os valores finais do atraso do filtro.*

3. Calcule $\boldsymbol{h[n]}$ analiticamente para o SLITD descrito pela equação de diferenças dada no item 1, e trace o gráfico de hastes (`stem`) desse sinal para $0 \le n \le 127$.

4. Crie um trem de impulsos centrado na origem com largura de 7 amostras com período de 128 amostras (considere $-63 \le n \le 64$). Esboce a resposta do SLITD do item 1 para o sinal trem de impulsos criado nesse item, usando as funções convolução e filtragem.

5. Determine as saídas do SLITD do item 1 para os sinais de entrada $\boldsymbol{x_1[n] = \cos(0{,}1\pi.n)}$ e $\boldsymbol{x_2[n] = \cos(0{,}9\pi.n)}$, para $0 \le n \le 127$. Explique as respostas observadas relacionando amplitudes e frequências.

Questões

1. Seja um SLITD somador, causal, em repouso (no domínio do tempo contínuo seria um *integrador*) – um sistema cuja saída no instante n, $y[n]$, é a soma da entrada $x[n]$ até o instante n (inclusive). Determine a função de transferência e a resposta ao impulso desse SLITD. Classifique também o sistema quanto ao tipo da resposta ao impulso.

$$y[n] = \sum_{k=-\infty}^{n} x[k]$$

2. Compare as respostas obtidas no item 5 dos procedimentos práticos com aquelas obtidas com o sistema em repouso descrito pela seguinte *função de transferência*, quando submetido às mesmas entradas.

$$H(z) = \frac{z}{(z-1)(z-1/2)}, \qquad |z|>1$$

Apêndice

Coeficientes e Denominação dos Filtros

A denominação dos filtros indicam a existência dos coeficientes $\boldsymbol{b(i)}$**'s** e $\boldsymbol{a(i)}$'s presentes na EDLCC (Equação 1) ou na Função de Transferência:

- Quando $N = 0$ (isto é, a é um escalar), o filtro é dito ser de *Finite Impulse Response* (FIR), e é chamado de filtro *all-zero* (só zeros), ou filtro não recursivo, ou filtro de média móvel (*Moving Average* - MA).

- Quando $M = 0$ (isto é, b é um escalar), o filtro é dito ser de *Infinite Impulse Response* (IIR), e é chamado de filtro *all-pole* (só polos), ou filtro recursivo, ou filtro auto-regressivo (*Auto-Regressive* - AR).

- Se ambos N e M são maiores que zero, o filtro é dito ser de IIR, e é conhecido por filtro pole-zero, ou filtro recursivo, ou filtro de média móvel auto-regressivo (ARMA).

As abreviações AR, MA, e ARMA são usuais com filtros de processos estocásticos.

Função de Transferência

A função de transferência é a razão entre as transformadas (Laplace ou Z) da saída pela entrada de um sistema linear invariante no tempo. É uma representação básica de um filtro digital no domínio z, expressando o filtro como uma razão de dois polinômios em z. Ela é o principal modelo de representação de SLITDs (Equação 4).

Filtros e Funções de Transferência

Em geral, a Transformada-Z da saída $y[n]$ de um filtro digital, $Y(z)$, e pode ser relacionada à Transformada-Z da entrada $x[n]$, $X(z)$, transformando-se para z a Equação (1):

$$Y(z) = H(z)X(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{a_1 + a_2 z^{-1} + \cdots + a_N z^{-N}} X(z)$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{a_1 + a_2 z^{-1} + \cdots + a_N z^{-N}} \qquad (4)$$

onde $H(z)$ é a função de transferência (função rede, função sistema) do filtro. As constantes $\mathbf{b_k}$**'s** e $\mathbf{a_k}$'s são os coeficientes do filtro e a ordem do filtro é o maior valor entre N e M. O Matlab armazena os coeficientes do numerador e do denominador em dois vetores linha.

Obs.: Os índices de vetores no Matlab são indexados a partir do índice 1, e no Python a partir de 0.

Matlab:

```
x = …;                  % sinal de entrada ao SLIT
b = [b0 b1 … bM];       % coef.s que multiplicam as entradas
b = [a0 a1 … aN];       % coef.s que multiplicam as saídas
y = filter(b,a,x);      % filtragem (saída do SLIT)
```

Python:

```
Import numpy as np
Import scipy.signal as ss
x = …;                          # sinal de entrada ao SLIT
b = np.array([b0, b1,… ,bM])    # coef.s que multiplicam as entradas
b = np.array([a0, a1,… ,aN])    # coef.s que multiplicam as saídas
y = ss.lfilter(b,a,x)           # filtragem (saída do SLIT)
```