國立臺北科技大學
National Taipei University of Technology

數位信號處理實習

LAB2

電子工程系 蔡偉和 教授

100360318 四子三甲 陳奕璋 學生

2014/3/17

[練習 2-1] 不使用內建函式 conv()而改以迴圈累加方式來實現 Convolution。

1. 程式碼：

使用 For 迴圈實現

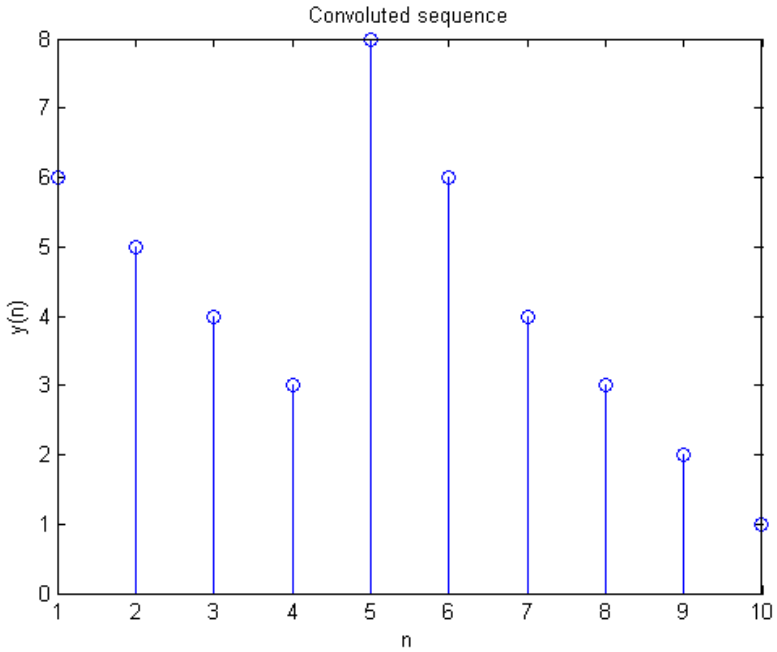
```
% Implement convolution with for loop

A = [6 5 4 3 2 1];
B = [1 0 0 0 1];

Al = length(A);
Bl = length(B);

N = 1 : Al+Bl-1;
Nl = length(N);

y = zeros(1, Nl);

for i = N
    for j = 1 : Al
        if((i+1-j) > 0)
            if((i+1-j) > Bl)
              y(i) = y(i);
            else
              y(i) = y(i) + A(j) * B(i+1-j);
            end
        end
    end
end

disp(y);
stem(N, y);
title('Convoluted sequence'); xlabel('n'); ylabel('y(n)');
```

使用矩陣實現

```
% Implement convolution with matrix

A = [6 5 4 3 2 1];
B = [1 0 0 0 1];

Al = length(A);
Bl = length(B);

N = 1 : Al+Bl-1;
Nl = length(N);

y = zeros(Bl, Nl);
```

```
for i = 1 : Bl
   for j = 1 : Al
       y(i, i+j-1) = B(1, i) * A(1, j);
   end
end

y = sum(y);

disp(y);
stem(N, y);
title('Convoluted sequence'); xlabel('n'); ylabel('y(n)');
```

2. Matlab 波形圖：



3. 實驗結果正確性

若使用 Matlab 內建之 conv 函數運算，和以上使用迴圈與矩陣運算的結果一致，可證明運算結果無誤。

執行 conv([6 5 4 3 2 1], [1 0 0 0 1])，可得以下結果，證明上述程式碼正確。

```
>> conv([6 5 4 3 2 1], [1 0 0 0 1])

ans =

     6     5     4     3     8     6     4     3     2     1
```

4. 心得

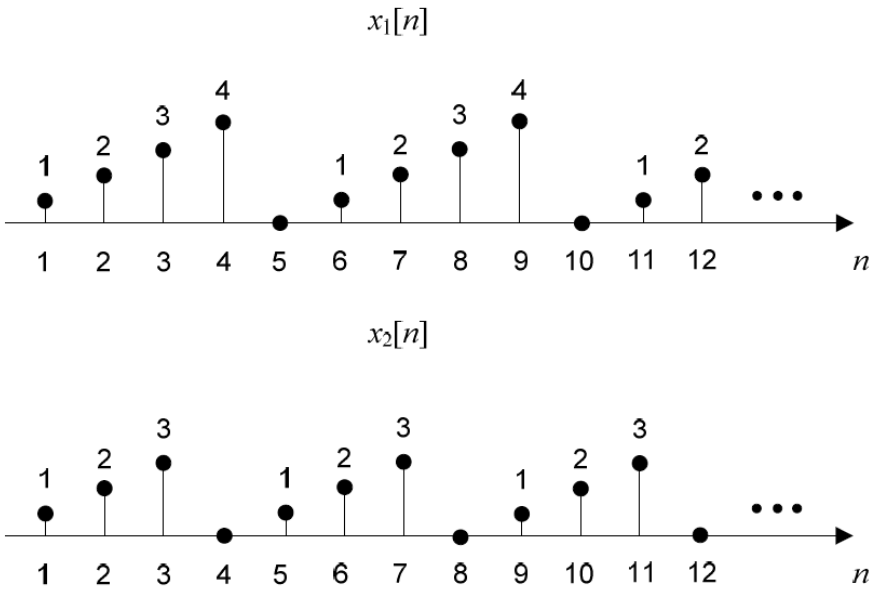
使用迴圈來實現 Convolution，加深了自己在 Matlab 的 for 迴圈熟練度，也練習到在 Matlab 兩層迴圈的寫法。

我覺得其實難的不是語法上的問題，而是要如何把數學式 $y[n] = \sum_{k=-\infty}^{\infty} x_1[k]x_2[n-k]$ 轉為程式碼實現，而單單轉換過去也不能執行，因為 Matlab 向量的起始值是從 1 開始，且不能有負值，因此還要加上 if 的判斷式來處理。

在此次實驗還學習到了下列函數。

1. `zeros(m, n)`
   可傳回一個初始化各元素維 0 的 m * n 矩陣。
2. `length(a)`
   可回傳 a 矩陣之 n columns 之長度。
3. `disp(x)`
   顯示 x 變數之內容。

[練習 2-2] 若有兩訊號 $x_1[n] = n \% 5$, $x_2[n] = n \% 4$, $1 \le n \le 1000$



試利用矩陣運算方式來實現 Convolution，並比較此方式與上題迴圈累加方式的運算速度差別。

1. 程式碼：

使用 For 迴圈實現

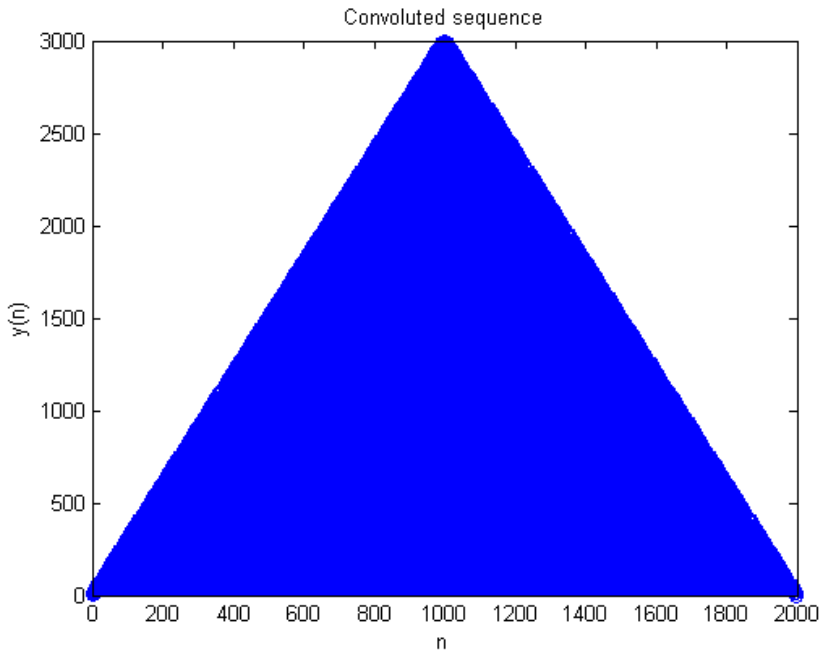
```
% Duplicate convolution with for loop

% make matrix
basisN = 1000;
basisA = [1 2 3 4 0];
basisB = [1 2 3 0];

A = repmat(basisA, 1, basisN/length(basisA));
B = repmat(basisB, 1, basisN/length(basisB));

% calc started
Al = length(A);
Bl = length(B);

N = 1 : Al+Bl-1;
Nl = length(N);

y = zeros(1, Nl);

for i = N
    for j = 1 : Al
        if((i+1-j) > 0)
            if((i+1-j) > Bl)
              y(i) = y(i);
```

```
            else
              y(i) = y(i) + A(j) * B(i+1-j);
            end
        end
    end
end

disp(y);
stem(N, y);
title('Convoluted sequence'); xlabel('n'); ylabel('y(n)');
```

使用矩陣實現

```
% Duplicate convolution with matrix

% make matrix
basisN = 1000;
basisA = [1 2 3 4 0];
basisB = [1 2 3 0];

A = repmat(basisA, 1, basisN/length(basisA));
B = repmat(basisB, 1, basisN/length(basisB));

% calc started
Al = length(A);
Bl = length(B);

N = 1 : Al+Bl-1;
Nl = length(N);

y = zeros(Bl, Nl);

for i = 1 : Bl
   for j = 1 : Al
       y(i, i+j-1) = B(1, i) * A(1, j);
   end
end

y = sum(y);

disp(y);
stem(N, y);
title('Convoluted sequence'); xlabel('n'); ylabel('y(n)');
```

2. Matlab 波形圖：



3. 實驗結果正確性

使用內建的 conv 函數確定實驗結果正確性，執行下列程式碼，發現與上述程式會出之圖片一樣，可見實驗結果正確。

```
A = repmat([1 2 3 4 0], 1, 1000/5);
B = repmat([1 2 3 0], 1, 1000/4);
stem(1:1999, conv(A, B));
```

4. 比較

```
% Runtime comparison of forloop and matrix convolution implementation

clear;

N = 100;
eltime_forloop = zeros(1, N);
for ii = 1:N
    tic;
    evalc('lab02_2_forloop()');
    eltime_forloop(ii) = toc;
end

eltime_matrix = zeros(1, N);
```

```
for ii = 1:N
    tic;
    evalc('lab02_2_matrix()');
    eltime_matrix(ii) = toc;
end

avg_forloop = mean(eltime_forloop);
avg_matrix = mean(eltime_matrix);

fprintf('Average elapsed time of forloop one was %f seconds.\n', avg_forloop);
fprintf('Average elapsed time of matrix one was %f seconds.\n', avg_matrix);
fprintf('Matrix one was %f times faster then forloop one.\n',
         avg_forloop / avg_matrix);
```

執行上方程式後，可得下方執行結果，可得知矩陣的方法比迴圈的方法還快了 3.215495 倍。

```
Average elapsed time of forloop one was 0.283228 seconds.
Average elapsed time of matrix one was 0.088082 seconds.
Matrix one was 3.215495 times faster then forloop one.
```

5. 心得

使用矩陣實現與迴圈實現的思維不一樣，需要思考一陣子之後才知道要如何實現。

因為要比較兩者運算速度的差別，使用到 tic 與 toc 函數配合計算程式執行時間，因為取得兩者的執行時間比較沒有比較的依據，因此將兩者相除，計算出兩者何者較快幾倍，較有比較上的意義。

為了使結果的誤差更低，兩者各計算了 100 次，取執行時間的平均秒數。

在此次實驗學習到了下列函數。

1. `repmat(A, m, n)`
   回傳將 A 矩陣重複 m*n 後的新矩陣
2. `tic, toc`
   配合 tic 與 toc 可知在兩者中間程式碼執行的使用時間。
3. mean(m)
   將 m 矩陣取算術平均數。
4. fprintf
   與 C 語言的 printf 類似用法。