專題: MATLAB Programming Techniques

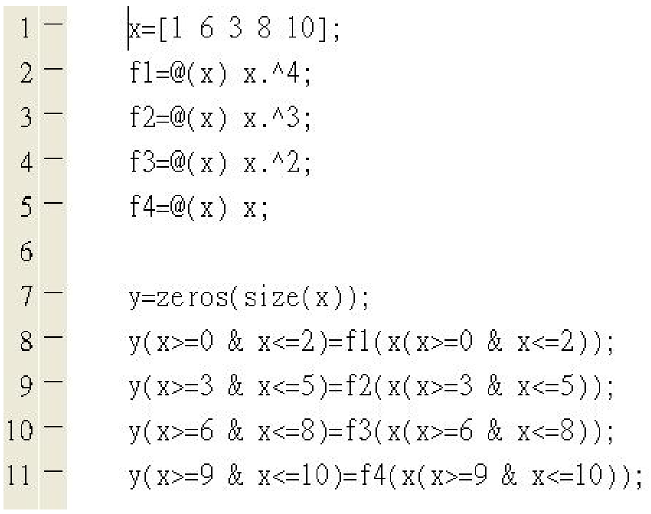
Problem 1:

Objective: The indexing operation for the MATLAB matrix manipulation plays an important role in terms of efficiency and elegancy.

For the function

$$f(x) = \begin{cases} x^4 & 0 \leq x \leq 2 \\ x^3 & 3 \leq x \leq 5 \\ x^2 & 6 \leq x \leq 8 \\ x & 9 \leq x \leq 10 \end{cases} \tag{1}$$

write a function or a short program such that when given a nx1 vector x whose components are all in $[0, 10]$ generate a function vector $y = f(x)$, i.e. given $x = [1\ \ 6\ \ 3\ \ 8\ \ 10]$, generate $y = [1\ \ 36\ \ 27\ \ 64\ \ 10]$ according to (1). The indexing operation is a technique that needs no "for loop" nor "if,else" commands. 圖 shows the beauty.



```
x=[1 6 3 8 10];
f1=@(x) x.^4;
f2=@(x) x.^3;
f3=@(x) x.^2;
f4=@(x) x;

y=zeros(size(x));
y(x>=0 & x<=2)=f1(x(x>=0 & x<=2));
y(x>=3 & x<=5)=f2(x(x>=3 & x<=5));
y(x>=6 & x<=8)=f3(x(x>=6 & x<=8));
y(x>=9 & x<=10)=f4(x(x>=9 & x<=10));
```

圖 1: One of the usage of indexing operation.

Problem 2:

Objective: This problem is to demonstrate the importance of the matrix computation by MATLAB as compared to the usage of the "for" loop which might be the first thought coming into the mind. The difference in efficiency will be seen when this computation needs to be done for thousands of times and when the dimension of the probelm increases as it is the common scenario in Monte Carlo simulations.

Let $X = [\mathbf{x_1}\ \mathbf{x_2}\ \cdots \mathbf{x_n}]$ be a pxn data matrix, where p designates the dimension and n represents the sample size. We want to calculate the following.

$$d = \sum_{j=1}^{n}\sum_{k=1}^{n} exp(\frac{-1}{2}\big(\mathbf{x_j} - \mathbf{x_k})' S_n^{-1}(\mathbf{x_j} - \mathbf{x_k})\big) \tag{2}$$

where S_n^{-1} represents the unbiased sample covariance matrix and is expressed by

$$S_n^{-1} = n^{-1}\sum_{j=1}^{n}(\mathbf{x_j} - \bar{\mathbf{x}}_\mathbf{n})'(\mathbf{x_j} - \bar{\mathbf{x}}_\mathbf{n})$$

in which $\bar{\mathbf{x}}_\mathbf{n} = n^{-1}\sum_{j=1}^{n}\mathbf{x_j}$ is the sample mean vector. The data matrix X can be generated as independent random samples from normal dirtribution, e.g.

```
X=normrnd(0,1,p,n)
```

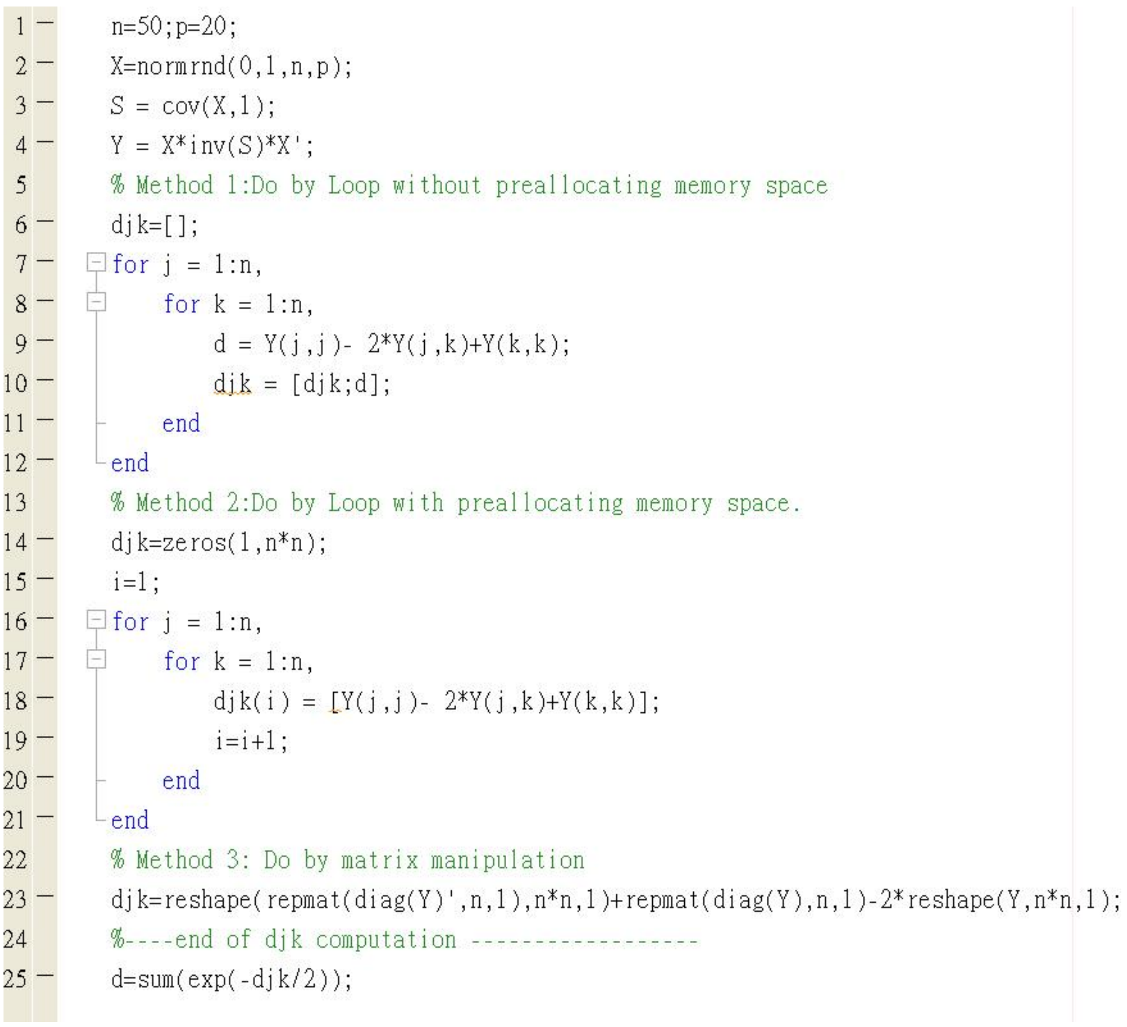
with $n = 50$ and $p = 2$.

1. Calculate d by using the technique of "for" loop.
2. Calculate d without using the technique of "for" loop.
3. Compare the time it takes for executing 10000 times of (a) and (b).

4. Compare the time it takes for executing 10000 times of (a) and (b) as p increases from 2 to 3,4,5,10 and 20.

5. Compare the time it takes for executing 10000 times of (a) and (b) as n varies from 10, 20, 50 to 100.

A demonstrated program for this problem is shown in 圖 2. You need to add some time-related instructions for each methd to collect the elapsed time.



```
n=50;p=20;
X=normrnd(0,1,n,p);
S = cov(X,1);
Y = X*inv(S)*X';
% Method 1:Do by Loop without preallocating memory space
djk=[];
for j = 1:n,
    for k = 1:n,
        d = Y(j,j)- 2*Y(j,k)+Y(k,k);
        djk = [djk;d];
    end
end
% Method 2:Do by Loop with preallocating memory space.
djk=zeros(1,n*n);
i=1;
for j = 1:n,
    for k = 1:n,
        djk(i) = [Y(j,j)- 2*Y(j,k)+Y(k,k)];
        i=i+1;
    end
end
% Method 3: Do by matrix manipulation
djk=reshape(repmat(diag(Y)',n,1),n*n,1)+repmat(diag(Y),n,1)-2*reshape(Y,n*n,1);
%----end of djk computation ------------------
d=sum(exp(-djk/2));
```

圖 2: Three methods for computing d in (2)

Problem 3:

Objective:The usage of the anonymous functions could be very elegant, neat and dramatic. The programmers should be acquainted with the techniques of applying anonymous functions in the right place. In most situations, the usage of the anonymous functions is straightforward. However the effect of combining anonymous functions with regular function could be huge.

Suppose two independent variables X and Y follow Beta distribution with their PDF being $f_X(x) = \beta(x|a_1, b_1)$, $f_Y(x) = \beta(x|a_2, b_2)$ respectively. Let a new variable Z be defined by

$$Z = XY$$

The variable Z does not look like following the Beta distribution. Now we want to approximate the distribution of Z by a beta distribution $\beta(a, b)$ in the following way

$$\min_{a,b} \int_0^1 \left(f(z) - \beta(z|a, b)\right)^2 \, dz \tag{3}$$

where $f(z)$ is the real distribution of Z and is written by

$$f(z) = \int_z^1 f_Y(y) f_X(z/y) \frac{1}{y} \, dy$$

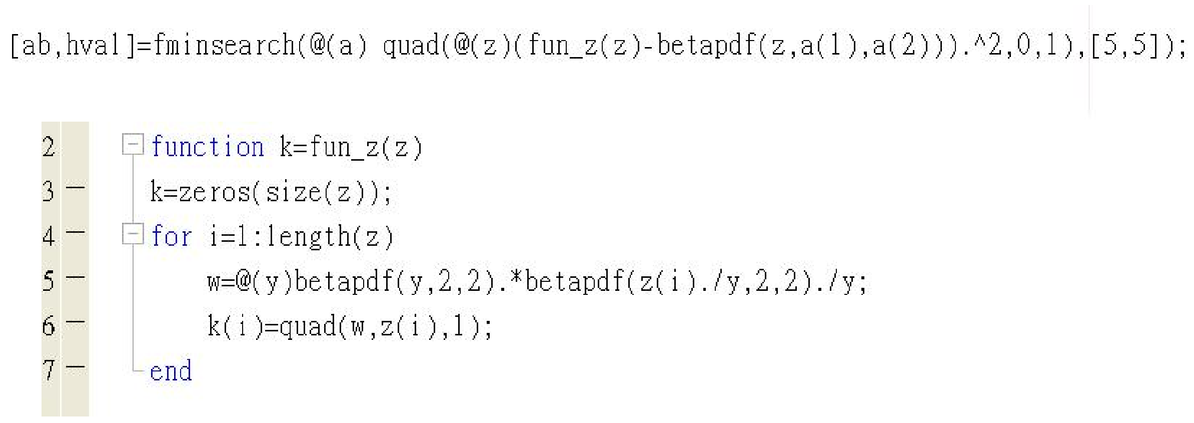
Let the parameters of the Beta functions $f_X(x)$ and $f_Y(x)$ be set to

$$a_1 = b_1 = a_2 = b_2 = 2$$

Now the problem is to solve the minimization problem (3) for the optimal parameters a and b.

To solve a multivariate optimization problem, MATLAB provides an instruction "fminsearch". The usage of "fminsearch" requires a function definition (the underlying function) as its first argument. In general, an anonymous

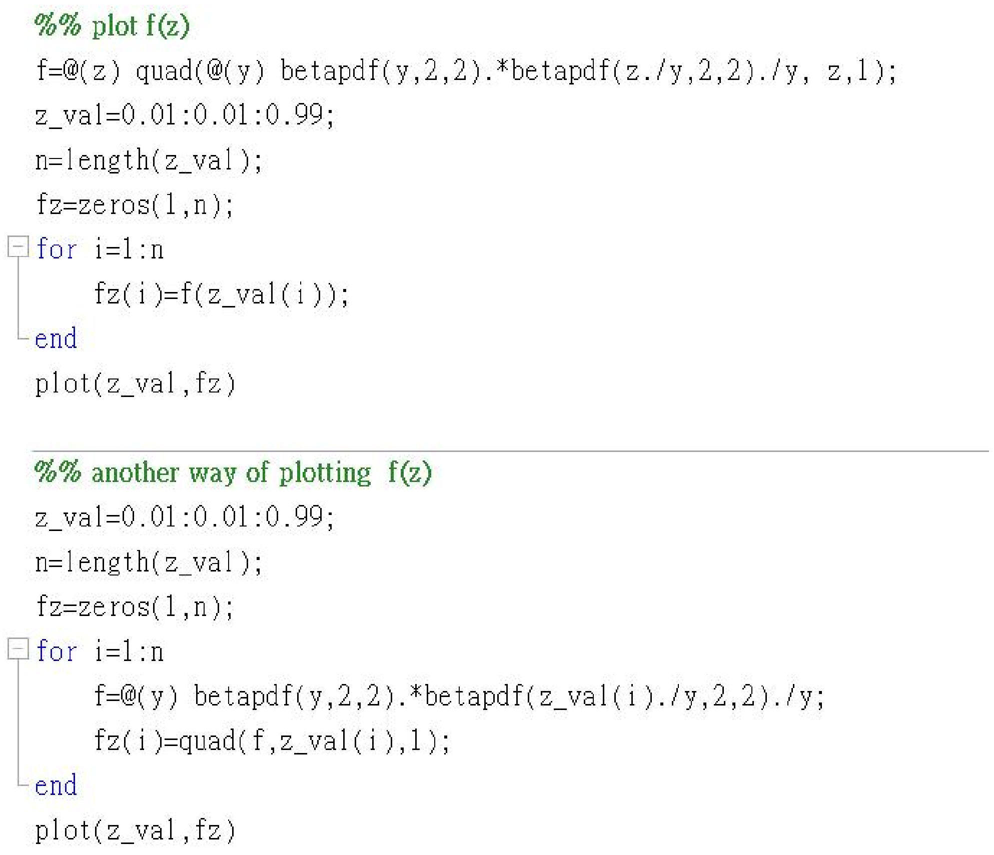
function is employed for this purpose. However, for complicated functions, a function file is necesary. 圖3 demonstrates the programs containing a main program (only one instruction) and a function (fun_z). This program is pretty concise and may not be easy to understand for inexperienced programmers. In fact, a complicated program starts from a simple one and finishes after a series of modifications. 圖 ??? shows another way of solving (3), although it is not condense, but works. The inexperienced programmers are suggested to draw $f(z)$ to get an idea of solving this problem. 圖 4 demonstrates the program of plotting $f(z)$ in two ways.
Meanwhile, it is a good idea to draw the figures of the real and the estimated distribution functions together to help check the correctness of the program or the goodness of the approach.



```
[ab,hval]=fminsearch(@(a) quad(@(z)(fun_z(z)-betapdf(z,a(1),a(2))).^2,0,1),[5,5]);
```

```
function k=fun_z(z)
k=zeros(size(z));
for i=1:length(z)
    w=@(y)betapdf(y,2,2).*betapdf(z(i)./y,2,2)./y;
    k(i)=quad(w,z(i),1);
end
```

圖 3: The usage of anonymous functions and regular function.



```
%% plot f(z)
f=@(z) quad(@(y) betapdf(y,2,2).*betapdf(z./y,2,2)./y, z,1);
z_val=0.01:0.01:0.99;
n=length(z_val);
fz=zeros(1,n);
for i=1:n
    fz(i)=f(z_val(i));
end
plot(z_val,fz)

%% another way of plotting  f(z)
z_val=0.01:0.01:0.99;
n=length(z_val);
fz=zeros(1,n);
for i=1:n
    f=@(y) betapdf(y,2,2).*betapdf(z_val(i)./y,2,2)./y;
    fz(i)=quad(f,z_val(i),1);
end
plot(z_val,fz)
```

圖 4: Two approaches of plotting $f(z)$.

Problem 4:
關於字串與文數字的轉換應用: 有一個混合常態的密度函數

$$f(x) = \pi_1 f(x|\mu_1, \sigma_1^2) + \pi_2 f(x|\mu_2, \sigma_2^2)$$

其中 $\pi_1 + \pi_2 = 1$, $f(x)$ 是常態密度函數。

1. 寫一支程式使用 ezplot 指令畫出上述函數, 且在程式執行時可以讓使用者自由輸入不同的參數。

2. 為方便模擬試驗, 需要產生服從上述分配的資料。假設 $\mu_1 = 40, \sigma_1 = 10, \mu_2 = 60, \sigma_2 = 20$ 固定不變, 改變混合比例 π_1 與樣本數 n 的值, 譬如 $\pi_1 = 0.1$:

$0.1 : 0.9$, $n = 50 : 50 : 300$, 在不同的 π_1 與 n 分別產生 $N = 10000$ 組樣本, 並儲存在特定目錄下。儲存時的檔名必須含 π_1 與 n 值以利辨識, 必要時可以加上日期。

3. 模擬時需要將資料從一個個檔案讀出來, 試著寫一段程式, 將上述的檔案依 π_1 與 n 之不同, 分別讀出並畫直方圖觀察。

4. 如果資料的產生需要從不同的分配組合而來, 也就是 $f(x)$ 是不同的分配函數, 這時候寫一個函數 (function) 來處理資料的生成是必要的。試著在資料產生時加入 $f(x)$ 是 T 與 卡方分配的選項。

5. 進行參數估計時, 依上述產生的檔案分別呼叫不同的演算法 (寫成 function 型態, 輸出爲參數估計值與概似函數值), 將估計結果寫在檔案上, 檔名的組合依上述原則。

6. 將上述在不同 π_1 與 n 組合下的 N 組估計結果計算適當的統計量, 譬如, 平均值、標準差或信賴區間, 並寫一支程式將結果表達成 Latex 的表格形式, 或畫出適當的圖形。

這個問題牽涉到文數字的轉換與字串的組合技巧。使用的指令參考: input, menu (固定選項時), num2str, str2num, strcat, clock (datevec)。亂數的產生必須選擇 seed, 建議下列指令

RandStream.setDefaultStream(RandStream('mt19937ar','seed',sum(100*clock)));
