Sample Solutions of Programming assignment of Midterm 1

Yu-Rong Wang and Cheng-Hsin Hsu

Note that, the solutions are for your reference only. If you have any doubts about the correctness of the answers, please let the instructor and the TA know. More importantly, like other math questions, the homework questions may be solved in various ways. Do not assume that the sample solutions here are the only *correct* answers; discuss with others about alternate solutions.

We will not grade your homework assignment, but you are highly encouraged to discuss with us during the Lab hours. The correlation between the homework assignments and quiz/midterm/final questions is high. So you do want to practice more and sooner.

1 Midterm Exam 1 (25 points + 1 bonus point)

2) $B_{(n+1)\times(n+1)} = \begin{bmatrix} A_{n\times n} & a_{n\times 1} \\ a^T_{1\times n} & \alpha_{1\times 1} \end{bmatrix}$

a) (i) Suppose $\alpha \leqslant 0$.
$\because B$ is positive definite.
$\therefore$ Let $y = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}^T_{n\times 1}$

$$\Rightarrow 0 < y^T B y = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}^T \begin{bmatrix} A & a \\ a^T & \alpha \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a^T & \alpha \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

$$= \alpha \quad \longrightarrow\!\!\!\!\!\!\times$$

(ii) Suppose A is not positive definite.
$\because B$ is positive definite.
$\therefore$ Let $z = \begin{bmatrix} v^T & 0 \end{bmatrix}$, where $v = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T_{n\times 1}$

$$\Rightarrow 0 < z^T B z = \begin{bmatrix} v^T & 0 \end{bmatrix} \begin{bmatrix} A & a \\ a^T & \alpha \end{bmatrix} \begin{bmatrix} v \\ 0 \end{bmatrix} = \begin{bmatrix} v^T A & v^T a \end{bmatrix} \begin{bmatrix} v \\ 0 \end{bmatrix}$$

$$= v^T A v \quad \longrightarrow\!\!\!\!\!\!\times$$

Therefore, by (i) and (ii), α must be positive and the $n \times n$ matrix A must be positive definite.

b) Let $A = LL^T$ and $l = L^{-1}a$

$$\Rightarrow \begin{bmatrix} L & O \\ l^T & \sqrt{\alpha - l^T l} \end{bmatrix} \begin{bmatrix} L^T & l \\ O & \sqrt{\alpha - l^T l} \end{bmatrix} = \begin{bmatrix} A & a \\ a^T & \alpha \end{bmatrix} = B$$

2 Computer Problem

1)

```
function midterm(n,t) % performance of algorithms for matrix
   multiplication
iterations = [1:t]'; sdotTime = zeros(t,1); saxpyTime = zeros(t,1);
disp('Time␣for␣using␣sdot:␣␣Time␣for␣using␣saxpy:');
for l=1:t
       A = rand(n,n); B = rand(n,n); C = zeros(n,n);
       sdotTime(l,1) = sdot(A, B, C, n);
       saxpyTime(l,1) = saxpy(A, B, C, n);
       fprintf('␣␣␣␣␣%f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣%f\n', sdotTime, saxpyTime);
end
plot(iterations, sdotTime, 'b-', iterations, saxpyTime, 'r--');
xlabel('iteration'); ylabel('time');
legend('Time␣for␣using␣sdot', 'Time␣for␣using␣saxpy');
end
%% sdot program
function sdotTime = sdot(A, B, C, n)
tic;
for i=1:n
       for j=1:n
              C(i,j) = A(i,:)*B(:,j);
       end
end
sdotTime = toc;
end
%% saxpy program
function saxyTime = saxpy(A, B, C, n)
tic;
for j=1:n
       C(:,j) = A*B(:,j);
end
saxyTime = toc;
end
```

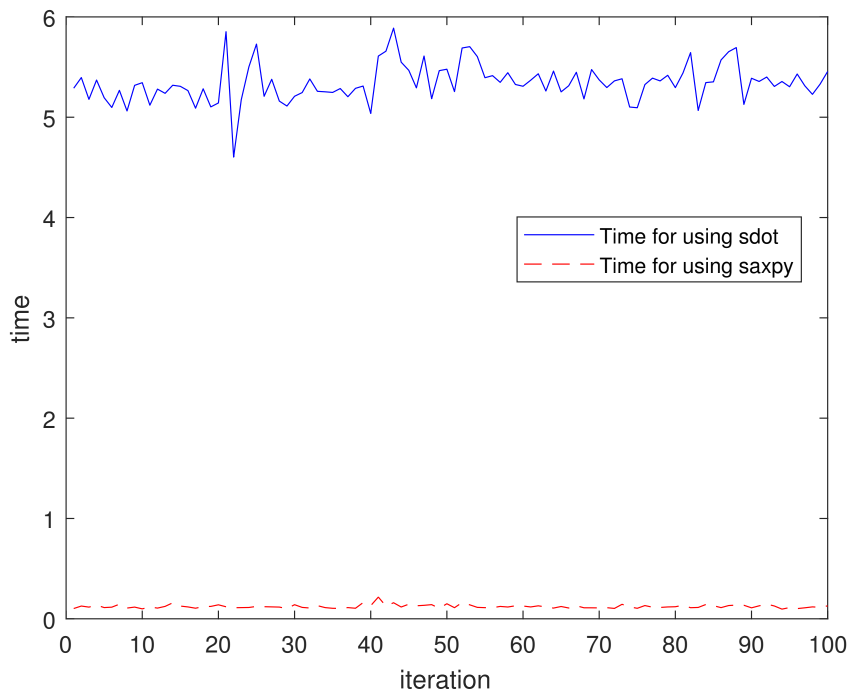

Figure 1: Time of sdot and saxpy

2) We implemented matrix multiplication C=AB 100 times, where A, B and C are 500x500 matrices and A and B comprise random entries(in [0, 1]). We plotted the time used by sdot and saxpy in Figure 1, and the following table shows the time and the ratio of the first ten times of the matrix multiplications.

Time for using sdot:	Time for using saxpy:	Ratio(sdot/saxpy):
5.289226	0.102398	51.653570
5.395224	0.127130	42.438571
5.178193	0.116464	44.461590
5.370549	0.136596	39.317103
5.194405	0.111824	46.451492
5.097250	0.115873	43.989927
5.267862	0.145511	36.202621
5.062432	0.107030	47.299100
5.317901	0.116621	45.600001
5.344271	0.099043	53.959345

3) We found that the time using saxpy is much less than the time using sdot. The reason is that a MATLAB matrix is stored (by columns) in contiguous locations in the computer's RAM, sdot which intent to compute each row of C takes much more time than that saxpy which intent to compute each column of C takes.