

# ROB 501 - Mathematics for Robotics

## HW #7

Due at 3PM on Wed, Oct. 20, 2021  
To be submitted on Canvas

- **Remark A)** One of the main focuses of this HW set is Recursive Least Squares (RLS), in other words, least squares estimation that can be implemented in a real-time environment. This is a good way to prepare for the Kalman Filter. The HW set looks long because lots of details are given for each step in a problem, which should make the work go quickly.
  - **Remark B)** To work the two problems on RLS, read the handouts `WeightedLeastSquares_RLS.pdf`, which is typeset and the handout `RecursiveLeastSquares_v02.pdf`, which is handwritten. These are under "Files/Handouts (Mostly Typeset by Grizzle on Individual Topics)" on Canvas. They are basically the same, but the handwritten one has a bit more detail. Read the handout and then implement the algorithm. This may be a bit tough on you, but when we go through it in lecture, you will be remarkably well prepared. The hints give you a LOT OF INFORMATION. Read them!
1. The symmetric matrix below has distinct e-values. Factor it as a product  $O\Lambda O^\top$  where  $O$  is an orthogonal matrix and  $\Lambda$  is diagonal. It is OK to use MATLAB.

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

2. The symmetric matrix below has repeated e-values  $2, 2, -1$ . In lecture, we stated that even with repeated e-values, we can still diagonalize a symmetric matrix using orthogonal matrices. The objective of the problem is to see why this is true by working a numerical example. We will follow the proof attached at the end of the HW set and factor  $A$  as a product  $O\Lambda O^\top$ , where  $O$  is an orthogonal matrix.

$$A = \begin{bmatrix} 1 & 0 & \sqrt{2} \\ 0 & 2 & 0 \\ \sqrt{2} & 0 & 0 \end{bmatrix}.$$

Each of the steps below is motivated by a step in the proof. **Suggestion:** Open a script file in MATLAB and execute each step of the problem. It will save time.

- (a) Verify that  $v^1 = [0, 1, 0]^\top$  satisfies  $Av^1 = 2v^1$ , and thus  $v^1$  is an e-vector corresponding to  $\lambda = 2$ .
- (b) Choose  $v^2$  and  $v^3$  such that  $\{v^1, v^2, v^3\}$  is orthonormal, and verify that  $V = [v^1 \ v^2 \ v^3]$  is an orthogonal matrix. In general, you would accomplish this by completing  $\{v^1\}$  to a basis of  $\mathbb{R}^n$  and applying Gram Schmidt. Here, you can do it by inspection.
- (c) Form the matrix  $V^\top AV$  and verify that it has the form

$$\begin{bmatrix} 2 & 0_{1 \times 2} \\ 0_{2 \times 1} & A_2 \end{bmatrix}$$

with  $A_2$  symmetric.

- (d) Use MATLAB to compute the e-values and e-vectors of  $A_2$ , and verify that they are 2,  $-1$ , and thus distinct<sup>1</sup>. Find  $U_2$  orthogonal such that

$$U_2^\top A_2 U_2$$

is diagonal. It is OK to use MATLAB for this.

- (e) Define the  $3 \times 3$  matrix  $U$  by

$$U = \begin{bmatrix} 1 & 0_{1 \times 2} \\ 0_{2 \times 1} & U_2 \end{bmatrix}.$$

Verify that  $U$  is orthogonal.

- (f) Define  $O = VU$  and verify that  $O$  is orthogonal.

- (g) **This is the only part you turn in:** Report what you get when you compute  $O^\top A O$ . If it is not diagonal, you have done something wrong.

3. Download the file `HW07_Data.zip` from the CANVAS MATLAB folder and load the file `DataHW07_Prob3.mat` into your MATLAB workspace (See also the `ReadMe.txt` file). The data file provides “perturbed or noisy” data for the model  $y_i = C_i x + e_i$ ,  $1 \leq i \leq N$ , where  $N = 500$ ,  $x \in \mathbb{R}^{100}$  and  $y_i \in \mathbb{R}^3$ . The data set contains the measured values  $y_i$ , the model matrices  $C_i$ , and the true value of  $x$ . The true value is given so that you can compare your estimated values to the true value. Of course, in real life, we would not have  $x$  available to us.

For  $1 \leq k \leq N$ , define  $S_k = I_{3 \times 3}$  and as in the lecture on Recursive Least Squares,

$$Y_k = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix}, \quad A_k = \begin{bmatrix} C_1 \\ \vdots \\ C_k \end{bmatrix}, \quad R_k = \text{diag}[S_1, \dots, S_k] = I.$$

- (a) Find  $n$  such that  $A_k$  has at least  $\dim(x) = 100$  independent columns for  $k \geq n$ . For each  $n \leq k \leq N$ , define

$$\hat{x}_k := \arg \min \|Y_k - A_k x\| = \arg \min \sqrt{(Y_k - A_k x)^\top R_k (Y_k - A_k x)}$$

but do not compute anything except  $n$  at this step.

- (b) For each  $n \leq k \leq N$ , compute  $\hat{x}_k$  in a batch process, that is,

$$\hat{x}_k = (A_k^\top R_k A_k)^{-1} A_k^\top R_k Y_k,$$

and, using the standard Euclidean norm, compute

$$E_k := \|\hat{x}_k - x\|.$$

Make a plot of  $E_k$  versus  $k$  and turn it in. Put a clear title on your plot, such as “Norm error in  $\hat{x}$  using Batch Process”. Implementing the “for each  $n \leq k \leq N$ ” will require a **for loop** or **while loop**. Use the **tic** and **toc** commands to determine how long it takes to compute your *entire set of estimates* and report this value. Either write it on your error plot by hand or place it there with a MATLAB command.

---

<sup>1</sup>If the e-values were repeated, you would find one e-vector, use it to build an orthonormal basis, and decompose  $A_2$  to find  $A_3$  symmetric that has dimension one less than  $A_2$ , etc.

- (c) For each  $n \leq k \leq N$ , compute  $\hat{x}_k$  using the RLS (Recursive Least Squares) Algorithm. First implement it without using the Matrix Inversion Lemma. Turn in a plot of  $E_k$  versus  $k$ , and record on your plot the amount of time it takes to do your computations.
- (d) For each  $n \leq k \leq N$ , compute  $\hat{x}_k$  once again using the RLS (Recursive Least Squares) Algorithm, but this time, implement it using the Matrix Inversion Lemma. Turn in a plot of  $E_k$  versus  $k$ , and record on your plot the amount of time it takes to do your computations. Note that this time you are numerically inverting a  $3 \times 3$  matrix and then computing the inverse of the  $100 \times 100$  matrix  $Q_k$  with the Matrix Inversion Lemma. This is the main point of the Matrix Inversion Lemma.
4. Download the file `HW07_Data.zip` from the CANVAS MATLAB folder and load the file `DataHW07_Prob4.mat` into your MATLAB workspace. It provides “perturbed or noisy” data for the model  $y_i = C_i x_i + e_i$ ,  $1 \leq i \leq N$ , where this time the “state” or “parameter”  $x$  that we are estimating is slowly “drifting” (means that it is slowly varying with time), which is why it has an index  $x_i$ . We will see that basic least squares does not work very well when  $x$  can drift. We will learn a way to fix it.

In this problem,  $N = 500$ ,  $x \in \mathbb{R}^{20}$  and  $y_i \in \mathbb{R}^3$ . The data set contains the measured values  $y_i$ , the model matrices  $C_i$ , and the true value of  $x_i$ . The true value is given so that you can compare your estimated values to the true value. As you know very well, in real life, we would not have  $x_i$  available to us.

- (a) Find  $n$  such that  $A_k$  has at least  $\dim(x) = 20$  independent columns for  $k \geq n$ . For each  $n \leq k \leq N$ , define

$$\hat{x}_k := \arg \min \|Y_k - A_k x\|,$$

but do not compute anything except  $n$  at this step. You should find  $n = 7$ .

- (b) As in Prob. 4, use constant weights, with  $S_k = I_{3 \times 3}$ . For each  $n \leq k \leq N$ , compute  $\hat{x}_k$  (any method you wish) and compute  $E_k := \|\hat{x}_k - x_k\|$ . It does not matter how fast your MATLAB code is for the computation of  $\hat{x}_k$  because in this problem we will not record the time. Make a plot of  $E_k$  versus  $k$  and turn it in. Put a clear title on your plot. Note that the error gets pretty bad.
- (c) **The forgetting factor:** Let  $0 < \lambda < 1$  (some number strictly between zero and one). A typical value for the *forgetting factor* might be  $\lambda = 0.98$ . The idea is to discount old measurements when we do the least squares problem. This is done by selecting at time  $k$  the weight matrices for  $1 \leq i \leq k$  to be

$$S_i = \lambda^{(k-i)} I_{3 \times 3}.$$

With this choice, the  $3k \times 3k$  weighting matrix  $R_k$  is given by

$$R_k = \text{diag}(\lambda^{k-1} I_3, \lambda^{k-2} I_3, \dots, \lambda I_3, I_3).$$

We see that the errors in older measurements are “discounted” by higher powers of  $\lambda$ , and thus the estimation process “exponentially forgets” them and “focuses” on the more recent measurements. It is important to note that at each step  $k$ , we are redefining the weights  $R_k$  so that errors in the newest measurements are penalized the most. This can be done recursively in our `for` loop, by

$$R_{k+1} = \begin{bmatrix} \lambda R_k & 0_{3k \times 3} \\ 0_{3 \times 3k} & I_{3 \times 3} \end{bmatrix}.$$

For each  $n \leq k \leq N$ , compute  $\hat{x}_k$  using the Batch Method. Turn in a plot of  $E_k$  versus  $k$ , and label your plot appropriately. You can use  $\lambda = 0.98$  or you can tune the forgetting factor to see what works best. How can you resist playing with it once you have your code working? :)

- (d) For each  $n \leq k \leq N$ , compute  $\hat{x}_k$  now using the RLS (Recursive Least Squares) Algorithm, with forgetting factor. The algorithm (without using the Matrix Inversion Lemma) becomes

- **Initialization Step:** Set

$$Q_n := \sum_{i=1}^n C_i^\top \lambda^{n-i} C_i$$

$$\Gamma_n := \sum_{i=1}^n C_i^\top \lambda^{n-i} y_i$$

$$\hat{x}_n := (Q_n)^{-1} \Gamma_n$$

- **Recursion:** For  $n \leq k < N$

$$Q_{k+1} := \lambda Q_k + C_{k+1}^\top C_{k+1}$$

$$K_{k+1} := (Q_{k+1})^{-1} C_{k+1}^\top$$

$$\hat{x}_{k+1} := \hat{x}_k + K_{k+1} (y_{k+1} - C_{k+1} \hat{x}_k)$$

- If you want the version with the Matrix Inversion Lemma, see the hints!
  - Turn in a plot of  $E_k := \|\hat{x}_k - x_k\|$  versus  $k$ , and label your plot appropriately. To be clear, there are no  $\lambda$ 's in the computation of  $E_k$ ; we are just using the standard Euclidean norm to see how well we are doing in tracking  $x$  as it slowly drifts.
5. Classify each matrix as positive definite, positive semi-definite, or neither. In addition, if the matrix is either positive definite or positive semi-definite, find a square root. You may use MATLAB to factor a symmetric matrix as  $\Lambda = O^\top P O$  or as  $P = O \Lambda O^\top$ .

(a)  $\textcircled{\smiley} = \begin{bmatrix} 1 & 3 \\ 3 & 9 \end{bmatrix}$ .

(b)  $\textcircled{\text{skull}} = \begin{bmatrix} 6 & 10 & 11 \\ 10 & 19 & 19 \\ 11 & 19 & 21 \end{bmatrix}$ .

(c)  $\textcircled{\frown} = \begin{bmatrix} 2 & 6 & 10 \\ 6 & 10 & 14 \\ 10 & 14 & 18 \end{bmatrix}$ .

6. Use the results on Schur Complements to solve the following problems by hand:

(a) Determine if  $\textcircled{\smiley} = \begin{bmatrix} 1 & 3 \\ 3 & 8 \end{bmatrix}$  is positive definite or not.

(b) Determine if  $\textcircled{\text{skull}} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 4 & 7 \\ 6 & 7 & 10 \end{bmatrix}$  is positive definite or not.

(c) Find the range of  $a$  such that the following matrix is positive definite:  $\textcircled{\frown} = \begin{bmatrix} 1 & 2 & 6 \\ 2 & 5 & 7 \\ 6 & 7 & a \end{bmatrix}$

7. Find  $x$  of minimum norm that satisfies the equation

$$\begin{bmatrix} 1 & 3 & 2 \\ 3 & 8 & 4 \end{bmatrix} x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

(a) Use the standard inner product on  $\mathbb{R}^3$

(b) Use the inner product  $\langle x, y \rangle = x^\top \begin{bmatrix} 5 & 1 & 9 \\ 1 & 2 & 1 \\ 9 & 1 & 17 \end{bmatrix} y$

## Hints

**Hints: Prob. 2** The important point here is that when a matrix is symmetric, repeated e-values do not pose a problem as they do for a general square matrix. The last page of the HW gives a proof by induction.

- (a) **Base Step:** The first thing to note is that a  $1 \times 1$  matrix can always be factored.
- (b) **Inductive Step:** The induction hypothesis is to assume that  $(n-1) \times (n-1)$  symmetric matrices can be factored as  $O\Lambda O^\top$  where  $O$  is an orthogonal matrix and  $\Lambda$  is diagonal.
- (c) **To show:** Next, you must show that the same is true for  $n \times n$  symmetric matrices. The key step in the proof is to show that if  $A$  is symmetric and  $\lambda$  is an e-value, then there exists an orthogonal matrix  $P$  such that

$$P^\top AP = \begin{bmatrix} \lambda & 0_{1 \times (n-1)} \\ 0_{(n-1) \times 1} & B \end{bmatrix},$$

where  $B$  is symmetric and  $(n-1) \times (n-1)$ . The orthogonal matrix  $P$  is produced by using an e-vector associated with  $\lambda$  and the Gram-Schmidt process. Hence, if you care to understand the proof, it is within your means to do so.

**Hints: Prob. 3** Recursive Least Squares (RLS)

- (a) **Basic Version:**

- Initialization Step: Choose  $n$  such that  $Q_n$  is invertible (full rank)

$$\begin{aligned} Q_n &:= \sum_{i=1}^n C_i^\top S_i C_i \\ \Gamma_n &:= \sum_{i=1}^n C_i^\top S_i y_i \\ \hat{x}_n &:= (Q_n)^{-1} \Gamma_n \end{aligned}$$

- Recursion: For  $n \leq k < N$

$$\begin{aligned} Q_{k+1} &:= Q_k + C_{k+1}^\top S_{k+1} C_{k+1} \\ K_{k+1} &:= (Q_{k+1})^{-1} C_{k+1}^\top S_{k+1} \\ \hat{x}_{k+1} &:= \hat{x}_k + K_{k+1} (y_{k+1} - C_{k+1} \hat{x}_k) \end{aligned}$$

- (b) **Improved Version Using the Matrix Inversion Lemma:**

- Initialization Step: Choose  $n$  such that  $Q_n$  is invertible (full rank)

$$\begin{aligned} Q_n &:= \sum_{i=1}^n C_i^\top S_i C_i \\ P_n &:= (Q_n)^{-1} \\ \Gamma_n &:= \sum_{i=1}^n C_i^\top S_i y_i \\ \hat{x}_n &:= P_n \Gamma_n \end{aligned}$$

- Recursion: For  $n \leq k < N$

$$\begin{aligned} P_{k+1} &= P_k - P_k C_{k+1}^\top [S_{k+1}^{-1} + C_{k+1} P_k C_{k+1}^\top]^{-1} C_{k+1} P_k. \\ K_{k+1} &:= P_{k+1} C_{k+1}^\top S_{k+1} \\ \hat{x}_{k+1} &:= \hat{x}_k + K_{k+1} (y_{k+1} - C_{k+1} \hat{x}_k) \end{aligned}$$

- How to Derive the Riccati Equation? It comes from the Matrix Inversion Lemma

$$\begin{aligned} Q_{k+1} &= Q_k + C_{k+1}^\top S_{k+1} C_{k+1} \\ Q_{k+1}^{-1} &= (Q_k + C_{k+1}^\top S_{k+1} C_{k+1})^{-1} \\ &= Q_k^{-1} - Q_k^{-1} C_{k+1}^\top [S_{k+1}^{-1} + C_{k+1} Q_k^{-1} C_{k+1}^\top]^{-1} C_{k+1} Q_k^{-1} \\ P_k &:= Q_k^{-1} \\ P_{k+1} &= P_k - P_k C_{k+1}^\top [S_{k+1}^{-1} + C_{k+1} P_k C_{k+1}^\top]^{-1} C_{k+1} P_k. \end{aligned}$$

- Jacopo Francesco Riccati (1676-1754) [http://en.wikipedia.org/wiki/Jacopo\\_Riccati](http://en.wikipedia.org/wiki/Jacopo_Riccati)

#### Hints: Prob. 4

- (a) Rewrite  $Q_{k+1} = \lambda Q_k + C_{k+1}^\top C_{k+1}$  as

$$\frac{1}{\lambda} Q_{k+1} = Q_k + C_{k+1}^\top \frac{1}{\lambda} C_{k+1}$$

Therefore

$$\lambda Q_{k+1}^{-1} = [Q_k + C_{k+1}^\top \frac{1}{\lambda} C_{k+1}]^{-1} \quad (*)$$

Using the Matrix Inversion Lemma, we have

$$\lambda Q_{k+1}^{-1} = Q_k^{-1} - Q_k^{-1} C_{k+1}^\top [\lambda I + C_{k+1} Q_k^{-1} C_{k+1}^\top]^{-1} C_{k+1} Q_k^{-1}.$$

and thus

$$Q_{k+1}^{-1} = \frac{1}{\lambda} Q_k^{-1} - \frac{1}{\lambda} Q_k^{-1} C_{k+1}^\top [\lambda I + C_{k+1} Q_k^{-1} C_{k+1}^\top]^{-1} C_{k+1} Q_k^{-1}.$$

If we define  $P_k := Q_k^{-1}$ , we obtain

$$P_{k+1} = \frac{1}{\lambda} P_k - \frac{1}{\lambda} P_k C_{k+1}^\top [\lambda I + C_{k+1} P_k C_{k+1}^\top]^{-1} C_{k+1} P_k.$$

- (b) If you are using your m-file for the Matrix Inversion Lemma, you can stop at (\*), apply your function to get the inverse of  $[Q_k + C_{k+1}^\top \frac{1}{\lambda} C_{k+1}]$ , and then divide by the forgetting factor.
- (c) The recursion on  $\hat{x}_k$  is unchanged from the RLS algorithm without the forgetting factor. In case you

want to see the derivation, the key formulas are:

$$\begin{aligned}
Q_k &:= \sum_{i=1}^k C_i^\top \lambda^{k-i} C_i \\
Q_k \hat{x}_k &:= \sum_{i=1}^k C_i^\top \lambda^{k-i} y_i \\
Q_{k+1} &= \sum_{i=1}^{k+1} C_i^\top \lambda^{k+1-i} C_i \\
&= \lambda Q_k + C_{k+1}^\top C_{k+1} \\
Q_{k+1} \hat{x}_{k+1} &= \sum_{i=1}^{k+1} C_i^\top \lambda^{k+1-i} y_i \\
&= \lambda \sum_{i=1}^k C_i^\top \lambda^{k-i} y_i + C_{k+1}^\top y_{k+1} \\
&= \lambda Q_k \hat{x}_k + C_{k+1}^\top y_{k+1} \\
\lambda Q_k &= Q_{k+1} - C_{k+1}^\top C_{k+1}
\end{aligned}$$

and thus, putting all of this together

$$\begin{aligned}
\hat{x}_{k+1} &= Q_{k+1}^{-1} [(Q_{k+1} - C_{k+1}^\top C_{k+1}) \hat{x}_k + C_{k+1}^\top y_{k+1}] \\
&= \hat{x}_k + Q_{k+1}^{-1} C_{k+1}^\top (y_{k+1} - C_{k+1} \hat{x}_k)
\end{aligned}$$

Hence, the only change is to the update formula for  $Q_{k+1}$ .

**Hints: Prob. 5** Use the `help eig` command in MATLAB.

**Hints: Prob. 6** Recall that for a symmetric matrix  $M = \begin{bmatrix} A & B \\ B^\top & C \end{bmatrix}$  the following are equivalent:

- (a)  $M \succ 0$
- (b)  $A \succ 0$  and  $C - B^\top A^{-1} B \succ 0$
- (c)  $C \succ 0$  and  $A - B C^{-1} B^\top \succ 0$

**Hints: Prob. 7** This is an under determined system of equations and not an over determined system of equations.



Reply With Quote

**Opalg**  
MHB Oldtimer

MHB Moderator

MHB Math Helper



**Status:** Offline

**Join Date:** Feb 2012

**Location:** Leeds, UK

**Posts:** 1,480

**Thanks:** 481 times

**Thanked:** 4510 times

**Awards:** 

DECEMBER 23RD, 2012, 12:21

#3

Originally Posted by **matqkks**

I have been trying to prove the following result:  
If  $A$  is real symmetric matrix with an eigenvalue  $\lambda$  of multiplicity  $m$  then  $\lambda$  has  $m$  linearly independent e.vectors.  
Is there a simple proof of this result?

This is a slight variation of Deveno's argument. I will assume you already know that the eigenvalues of a real symmetric matrix are all real.

Let  $A$  be an  $n \times n$  real symmetric matrix, and assume as an inductive hypothesis that all  $(n-1) \times (n-1)$  real symmetric matrices are diagonalisable. Let  $\lambda$  be an eigenvalue of  $A$ , with a normalised eigenvector  $x_1$ . Using the Gram-Schmidt process, form an orthonormal basis  $\{x_1, x_2, \dots, x_n\}$  with that eigenvector as its first element.

Let  $P$  be the  $n \times n$  matrix whose columns are  $x_1, x_2, \dots, x_n$ , and denote by  $T: \mathbb{R}^n \rightarrow \mathbb{R}^n$  the linear transformation whose matrix with respect to the standard basis is  $A$ . Then  $P$  is an orthogonal matrix ( $P^T = P^{-1}$ ), and the matrix of  $T$  with respect to the basis  $\{x_1, x_2, \dots, x_n\}$  is  $P^T A P$ . The  $(i, j)$ -element of that matrix is  $(P^T A P)_{ij} = \langle A x_j, x_i \rangle$ . In particular, the elements in the first column are

$$(P^T A P)_{i1} = \langle A x_1, x_i \rangle = \langle \lambda x_1, x_i \rangle = \begin{cases} \lambda & (i = 1) \\ 0 & (i > 1) \end{cases}$$

(because the vectors  $x_i$  are orthonormal). Thus the first column of  $P^T A P$  has  $\lambda$  as its top element, and 0 for each of the other elements. Since  $P^T A P$  is symmetric, the top row also consists of a  $\lambda$  followed by all zeros. Hence the matrix  $P^T A P$  looks like this:

$$\begin{bmatrix} \lambda & 0 & \dots & 0 \\ 0 & & & \\ \vdots & & B & \\ 0 & & & \end{bmatrix},$$

where  $B$  is an  $(n-1) \times (n-1)$  real symmetric matrix. By the inductive hypothesis,  $B$  is diagonalisable, so there is an orthogonal  $(n-1) \times (n-1)$  matrix  $Q$  such that  $Q^T B Q$  is

diagonal. Let

$$R = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & & & \\ \vdots & & Q & \\ 0 & & & \end{bmatrix}.$$

Then  $R^T P^T A P R$  is diagonal, as required.

**Was sich überhaupt sagen lässt, lässt sich klar sagen; und wovon man nicht reden kann, darüber muss man schweigen.**

(Anything that can be said at all, can be said clearly; and whereof one cannot speak, thereon one must be silent.)

– Ludwig Wittgenstein's good advice for forum contributors, in *Tractatus Logico-Philosophicus*.

[Reply With Quote](#)

[▲ Go to First Post](#)

« Singular Value decomposition | An inverse of the adjoint »

#### Similar Threads

[SOLVED] Incorporate axis symmetric case

By dwsmith in forum Mathematics Software and Calculator Discussion

Replies: 0

Last Post: December 6th, 2012, 14:56

Inverse of a Symmetric Matrix

By OhMyMarkov in forum Linear and Abstract Algebra

Replies: 2

Last Post: October 6th, 2012, 11:28

[SOLVED] Eigenvalues

By Sudharaka in forum Pre-Algebra and Algebra

Replies: 2

Last Post: May 27th, 2012, 09:02

Matrix Theory...showing that matrix is Unitary

By cylv89 in forum Linear and Abstract Algebra

Replies: 3

Last Post: March 27th, 2012, 17:00

A and B are two symmetric matrices

By Yankel in forum Linear and Abstract Algebra

Replies: 4

Last Post: January 27th, 2012, 09:17

-- Math Help Boards

Have a small screen? Select Math Help Boards for Small Screens!

Powered by vBulletin Copyright © 2000 - 2012, Jelsoft Enterprises Ltd.

Search Engine Optimisation provided by DragonByte SEO v1.0.15 (Pro) - vBulletin Mods & Addons Copyright © 2014 DragonByte Technologies Ltd.

Feedback Buttons provided by Advanced Post Thanks / Like (Pro) - vBulletin Mods & Addons Copyright © 2014 DragonByte Technologies Ltd.

© 2012-2014 Math Help Boards

[FORUMS](#)

[RULES](#)

[POTW](#)

[CONTACT](#)

[TOP](#)

