Rob 501 - Mathematics for Robotics
HW #6

Prof. Grizzle

Due Oct. 25, 2018 (Yes, no HW the week of Fall Break)
3PM via Gradescope[1]

- **Remark A)** There will be no HW due the week of Exam 1.
- **Remark B)** While the initial problems are on matrices, the main focus 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 C)** 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. 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 Λ 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.

[1] Some of you still have not understood that GradeScope is purposely set to allow a 3 hour grace period. Hence, if you submit by 5:59 PM, you are fine; there is no late penalty. At 6:01 PM you are locked out. At 6:00 PM exactly, I have no idea what happens!

(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[2]. 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 3×3 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 $O^\top AO$. If it is not diagonal, you have done something wrong.

3. Write an `m-file` or `function` in MATLAB[3] that implements the Matrix Inversion Lemma

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}.$$

Your function should assume that A^{-1} is provided, that is, it should not compute the inverse of the matrix A, however it should compute the inverse of C. Verify that your function works by using the data in Prob. 4 of HW # 5. **Turn in** your MATLAB code (print it out and insert it with your HW solutions).

4. Download the file `HW06_Data.zip` from the CANVAS MATLAB folder and load the file `DataHW06_Prob4.mat` into your MATLAB workspace (See also the ReadMe.txt file). The data file provides "perturbed or noisy" data for the model $y_i = C_i x + e_i$, $1 \le i \le N$, 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 $1 \le k \le N$, define $S_k = I_{3\times 3}$ and as in the lecture on Recursive Least Squares,

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

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

[2]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.

[3]It is OK to use a different programming language but please add enough comments that someone who does not know the language can still read it and see that it probably works.

(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 x-hat 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.

(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 3×3 matrix and then computing the inverse of the 100×100 matrix Q_k with the Matrix Inversion Lemma. This is the main point of the Matrix Inversion Lemma.

5. Download the file `HW06_Data.zip` from the CANVAS MATLAB folder and load the file `DataHW06_Prob5.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 $S_k = I_{3\times3}$. 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

$$S_i = \lambda^{(k-i)} I_{3\times3}.$$

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

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

We see that the errors in older measurements are "discounted" by higher powers of λ, 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 $n \le k \le N$, 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 $n \le k \le N$, 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

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

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

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

- 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 λ'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.

6. **Associating data to categories using distance:** Data association is a common task across various fields of Robotics. For example, if you have several pings from a sonar or radar, how do you know with which of several objects to associate them? You might enjoy the following YouTube video on the topic `https://www.youtube.com/watch?v=DK1DIcPwCOU`.

While doing data association, we look at the distance between a given point (or sample of data) and the mean (i.e. average value) of a finite set of possible categories, $S = \{S_1, S_2, \cdots, S_N\}$. The category assignment of a point x, called $C(x)$, is given by,

$$C(x) = \arg\min_{1\le i\le N} \; d(x, \mu_{S_i}),$$

where μ_{S_i} is the mean of category $S_i \in S$. The most common distance used is the Euclidean distance,

$$d_{Euc}(x, \mu_{S_i}) := ||x - \mu_{S_i}||_2$$

where μ_{S_1} is the mean of category S_i and $||x - \mu_{S_i}||_2$ is the standard 2-norm.

While Euclidean distance works well as a coarse estimator, sometimes we have more information than just the mean of each category. One such form of information is the covariance matrix, which gives the uncertainty along every direction of set. A large value of covariance in a direction indicates a larger uncertainty along that direction. Conversely, a smaller covariance value indicates more confidence along that direction.

For example, consider a category S_k with mean $\mu_{S_k} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and covariance $\Sigma_{S_k} = \begin{bmatrix} 1 & 0 \\ 0 & 9 \end{bmatrix}$. This can be represented as $S_k \sim N\left(\mu_{S_k}, \Sigma_{S_k}\right)$, the normal distribution with mean vector μ_{S_k} and covariance matrix Σ_{S_k}. The covariance along the x-axis is $\Sigma^{11}_{S_k} = 1$, which indicates that the points in this category have x values that are closer to the $\mu^x_{S_k}$ when compared to the y values and $\mu^y_{S_k}$ (since $\Sigma^{22}_{S_k} = 9$)

When the covariance matrix is available, we use a weighted 2-norm called the Mahalanobis distance. The Mahalanobis distance between a point and a category is defined as

$$d_M(x, \mu_{S_i}) := ||x - \mu_{S_i}||_{\Sigma^{-1}} = \sqrt{(x - \mu_{S_i})^T \Sigma^{-1}_{S_i} (x - \mu_{S_i})},$$

where μ_{S_1} is the mean and Σ_{S_i} is the covariance matrix of category i respectively. We recognize that we are using

$$< x, y >:= x^T \Sigma^{-1}_{S_i} y$$

, which gives the norm as $||x|| = \sqrt{x^T \Sigma^{-1}_{S_i} x}$, as discussed in lecture.

Given 2 categories,

$$S_1 \sim N\left(\begin{bmatrix} 4 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.25 & 0 \\ 0 & 2 \end{bmatrix}\right), \quad S_2 \sim N\left(\begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 6 & 2 \\ 2 & 2 \end{bmatrix}\right)$$

Find the association of the following points using (a) Euclidean and (b) Mahalanobis distance

$$(i)\ P_1 = \begin{bmatrix} 4 \\ 2 \end{bmatrix} \quad (ii)\ P_2 = \begin{bmatrix} 2.8 \\ 1 \end{bmatrix} \quad (iii)\ P_3 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

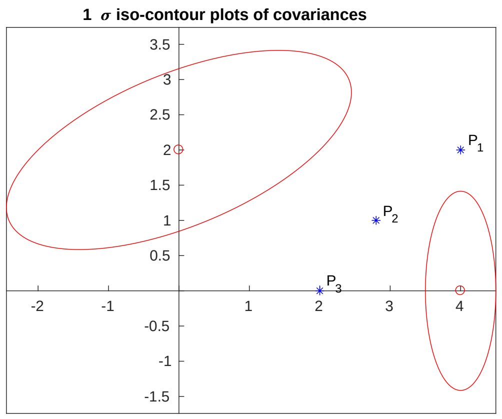


Figure 1: Red ellipses are points that have $d_M = 1$ from means of S_1 and S_2 with corresponding covariances.

Remark: Note that the covariance matrices here are positive definite matrices. As mentioned above, we encountered this kind of matrix norm when we covered weighted least squares. The inverse of a Covariance matrix is called an Information matrix. We will be learning more about this when we do Kalman filters. (And a lot more in **EECS 568** - Mobile Robotics)

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 1×1 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 Λ 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 λ is an e-value, then there exists an orthogonal matrix P such that

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

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

Hints: Prob. 4 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 $n \le k < N$

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

(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 $n \leq k < N$

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

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

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

- Jacopo Francesco Riccati (1676-1754) `http://en.wikipedia.org/wiki/Jacopo_Riccati`

Hints: Prob. 5

(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

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

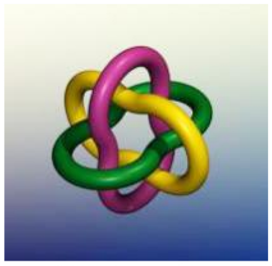
Hence, the only change is to the update formula for Q_{k+1}.

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 λ be an eigenvalue of A, with a normalised eigenvector x_1. Using the Gram–Schmidt process, form an orthonormal basis $\{x_1,x_2,\ldots,x_n\}$ with that eigenvector as its first element.

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

$$(P^{\mathrm{T}}AP)_{i1}=\langle Ax_1,x_i\rangle=\langle\lambda x_1,x_i\rangle=\begin{cases}\lambda & (i=1)\\ 0 & (i>1)\end{cases}$$

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

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

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 $Q^{\mathrm{T}}BQ$ is diagonal. Let

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

Then $R^{\mathrm{T}}P^{\mathrm{T}}APR$ 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 cylers89 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!

FORUMS RULES POTW CONTACT TOP

MATH HELP BOARDS.COM