MATH524 Midterm Exam: Solution by Zino Meyer ID 132611593

October 16, 2024, Wednesday, due 11:59PM the next day

1. **R for space-time data matrix**

   Use R to read the NOAAGlobalT.csv data file, and Select the four grid boxes that cover the following four locations: San Francisco (USA), Chicago (USA), London (UK), and Tokyo (Japan). Generate a 4×6 space-time data matrix for the June mean surface air temperature anomaly data of the four selected grid boxes and the six years from 1991 to 1996.
   Hint: You can download the NOAA Global Surface Temperature (NOAAGlobalTemp.csv) dataset from this midterm site on Canvas. Use the homework solution as a reference.

```
# Read data
anomaly_data <- read.csv("data/NOAAGlobalT.csv", header = TRUE)
dim(anomaly_data)
# [1] 2592 1648

# Define cities and years for later use
cities <- c("San Francisco", "Chicago", "London", "Tokyo")
years <- 1991:1996

# Select 4 rows for the cities
cities_rows <- anomaly_data[c(1920, 1927, 2088, 1828), ]

# Extract june data columns (every 12th column starting with the 9th)
june_cols <- seq(9, ncol(cities_rows), by = 12)

# Get june data only
june_data <- cities_rows[, june_cols]

# Extract years 91-96
W <- june_data[112:117]

# Set row and column names
row.names(W) <- cities
colnames(W) <- years

print(data.frame(W))
#                X1991   X1992   X1993   X1994   X1995   X1996
```

```
# San Francisco -1.5714  1.7334 -0.4119 -0.1086 -0.5146  0.0023
# Chicago        2.0285 -1.4971 -1.0721  0.4447  1.3542 -0.4503
# London        -1.2829  1.2994  0.4849 -0.0177  0.3695  0.0444
# Tokyo          1.0840 -0.1687 -0.4509  0.2126 -0.4990 -0.1933
```

1. **Matrix visualization and SVD calculation**

   a. Use R command `image()` or another R command to visualize (i.e., to plot) the 4×6 space-time data matrix W in the previous problem. Please include your code and figure in the pdf file. Please search around and find a proper color scheme, e.g., `col=topo.colors(64)`, for your command `image()` so that your grid boxes are clearly shown. Please transpose your matrix if necessary to show space locations as rows and time stamps as columns in your figure. See Fig. 1 as an example. Please use the correct text for city names, years, and your name for this exam.

```
cities <- c("San Francisco", "Chicago", "London", "Tokyo")
years <- 1991:1996

par(mar = c(3, 7, 3, 2)) # margins
image(t(W), col = heat.colors(64), axes = FALSE)
axis(1, at = seq(0, 1, length.out = 6), labels = years)
axis(2, at = seq(0, 1, length.out = 4), labels = cities, las = 2)
title(main = "Visualization of Temperature Data Matrix by Zino Meyer")
box()

dev.off()
```

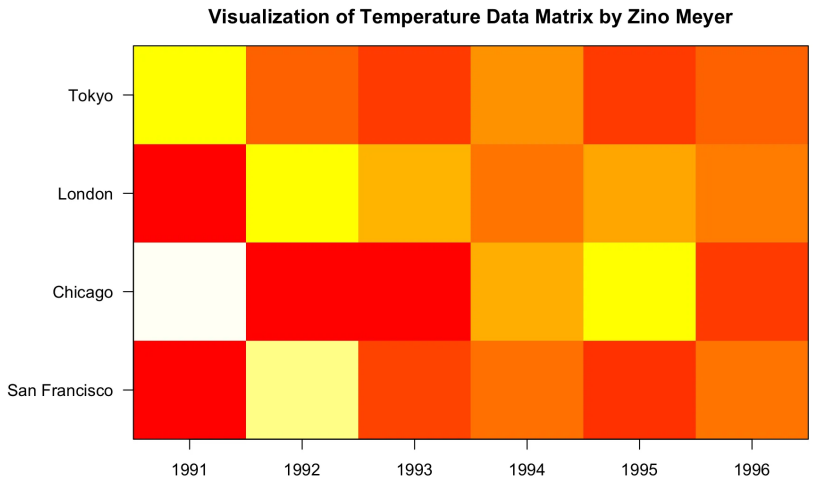

Figure 1: Visualization of Temperature Data Matrix

b. Use R to make an SVD analysis of the 4×6 matrix W.

```
W_svd <- svd(W)

U <- W_svd$u
U
# [,1]       [,2]       [,3]        [,4]
# [1,]  0.5309902 -0.2935757  0.7139986 -0.34938327
# [2,] -0.7094970 -0.6649731  0.2261012 -0.05747304
# [3,]  0.4113356 -0.5676375 -0.2050683  0.68303562
# [4,] -0.2132287  0.3865382  0.6301041  0.63881963

D <- W_svd$d
D
# [1] 4.206324 1.308161 1.219229 0.464540

V <- W_svd$v
V
# [,1]       [,2]       [,3]       [,4]
# [1,] -0.72092763  0.1984898  0.2319366  0.5352626
# [2,]  0.60695935 -0.2416753  0.4317358  0.5601015
# [3,]  0.19911408  0.2937743 -0.7546169  0.5353425
# [4,] -0.10122661 -0.1311811  0.1317200  0.2929954
# [5,] -0.23195046 -0.8806695 -0.3702604  0.0765775
# [6,]  0.09038491  0.1520006 -0.1895258 -0.1465547
```

c. Plot EOF1, i.e., the first column of the U matrix from the SVD analysis in Part (b), on a world map, as you did in your homework, but now you have four points instead of two points. You may use the dot size to represent the data values, and use red for positive and blue for negative. Or use your own way to illustrate EOF1 values.

```
library(maps)
library(mapdata)

EOF1 <- U[1, ]

# Combine data for easy access
locations <- data.frame(
    name = c("San Francisco", "Chicago", "London", "Tokyo"),
    lon = c(-122.4194, -87.6298, -0.1276, 139.6917),
    lat = c(37.7749, 41.8781, 51.5074, 35.6895),
    EOF_value = EOF1,
```

```
    pos = c(2, 1, 2, 4),
    offset = c(0.5, 0.7, 2, 0.6)
)

# Set color dependant on value
locations$cols <- ifelse(locations$EOF_value > 0, "red", "blue")

# Plot Mode 1
plot.new()
par(mar = c(0, 0, 0, 0))

map(database = "worldHires", ylim = c(-70, 70), mar = c(2, 2, 2, 2))
grid(nx = 12, ny = 6)

points(locations$lon, locations$lat,
    pch = 19,
    col = locations$cols,
    cex = (abs(locations$EOF_value) * 2)
    # point size is the absolute EOF value respectively (scaled by 2)
)

text(locations$lon, locations$lat,
    labels = paste(locations$name, round(locations$EOF_value, 2)),
    col = locations$cols,
    pos = locations$pos,
    offset = locations$offset,
    cex = 0.8
)

axis(1,
    at = seq(-180, 180, 60), col.axis = "black",
    tck = -0.05, las = 1, line = -0.9, lwd = 0
)
axis(2,
    at = seq(-70, 70, 20), col.axis = "black",
    tck = -0.05, las = 2, line = -0.9, lwd = 0
)

text(40, -55,
    "Surface Temperature Anomalies Mode 1",
    col = "purple", cex = 1.3
)
box()
```

```
dev.off()
```

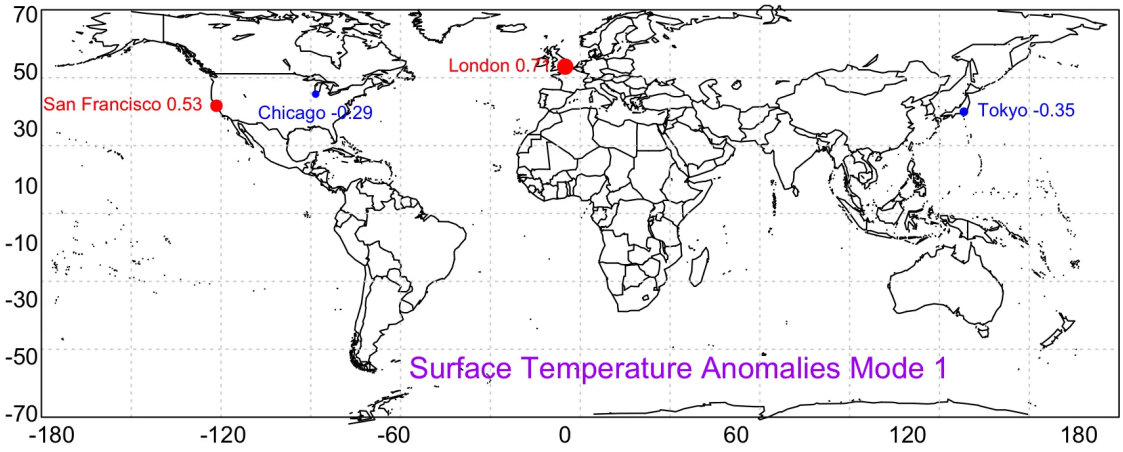

Figure 2: Surface Temperature Anomalies Mode 1. Red: positive, blue: negative

2. **Change a time-time data matrix to a data sequence, and then plot the data sequence**

   a. The data file EarthTemperatureData.csv contains the monthly and annual global average temperature anomalies from 1850 to 2015. Extract the monthly data from this data matrix and convert the monthly data into a sequence, i.e., a vector, according to time order: Jan 1850, Feb 1850, Mar 1850, ... , Dec 1850, Jan 1851, Feb 1881, ..., Nov 2015, Dec 2015.

```
et_data <- read.csv("data/EarthTemperatureData.csv", header = TRUE)
dim(et_data)
# [1] 166  14

monthly_data <- et_data[, 2:13]
monthly_seq <- c(t(monthly_data))
monthly_seq[1:7] # first seven elements in the array
# [1] -0.702 -0.284 -0.732 -0.570 -0.325 -0.213 -0.128
```

   b. Plot the temperature data sequence from Jan 1901 to Dec 2000. The horizontal axis should be marked as Year from 1901 to 2000. The vertical axis is for the temperature anomaly data. See

the solution for the sample midterm on Canvas for the figure style

```
monthly_seq_1901_2000 <- monthly_seq[600:1800]
time <- seq(1901, 2000, len = length(monthly_seq_1901_2000))

par(mar = c(5, 5, 2, 2))
plot(time, monthly_seq_1901_2000,
    type = "l", col = "blue", lwd = 2,
    xlab = "Time", ylab = "Temp anomalies [deg C]",
    main = "Monthly Global Average Temp Anomalies"
)

dev.off()
```

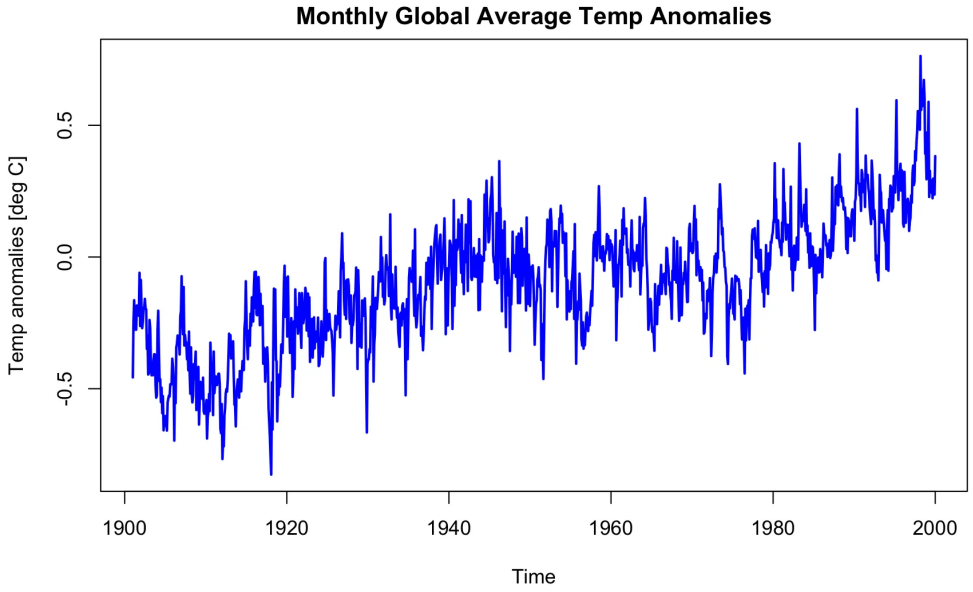

Figure 3: Monthly Global Average Temp Anomalies

3. **SVD theory**

   a. *State Theorem 1.1 on Page 23 of the textbook in your way.*

   Theorem 1.1 states that the covariance of a matrix X is expressed in its empirical orthogonal functions (EOFs), i.e. the spatial modes (U) of the SVD analysis of X. More specifically, it states that the eigenvector of the covariance matrix of X with the largest eigenvalue is the vector pointing in the direction where the data has the biggest variance. The eigenvector with the second largest eigenvalues points in the direction of the vector where the data has its second biggest variance, and so on. It furthermore states that the eigenvalues of the

covariance matrix are capture in the singular values of the SVD, where the eigenvalues are the square of the singular values divided by the number of columns of X.

b. *Prove the theorem. You may follow the proof in the textbook by filling in all the omitted details. Or you can generate your own proof that may be very different.*

Given:

$$A = UDV^t$$

$$C = \frac{1}{Y}AA^t$$

Wanted:

- Show that the eigenvectors $\mathbf{u}_k$ of the covariance matrix C are the same as the spatial modes from the SVD of A
- Show that the eigenvalues λ_k of the covariance matrix C relate to the SVD eigenvalues d_k of A by $\lambda_k = \frac{d_k^2}{Y}$

Proof:

1. Substituting A in C and rewriting:
   - Definition of Covariance Matrix

     $$C = \frac{1}{Y}AA^t$$

   - Substituting A in:

     $$C = \frac{1}{Y}(UDV^t)(UDV^t)^t$$

   - with $(UDV^t)^t = (V^t)^t D^t U^t = VDU^t$, we can rewrite by rearranging the parantheses:

     $$\begin{aligned} C &= \frac{1}{Y}(UDV^t)VDU^t \\ &= \frac{1}{Y}UD(V^tV)DU^t \end{aligned}$$

   - And since $V^tV = I_Y$, we can rewrite to:

$$C = \frac{1}{Y} UDIDU^t$$
$$= \frac{1}{Y} UD^2U^t$$

2. Substituting C in the eigenvalues problem:
   - We have the following problem:

$$C\mathbf{u}_k = \lambda_k \mathbf{u}_k$$

   - And with $C = \frac{1}{Y} UD^2U^t$, we can rewrite to:

$$\frac{1}{Y} UD^2U^tU = \lambda U$$
$$\frac{1}{Y} UD^2 = \lambda U$$

   - Thus:

$$\lambda = \frac{1}{Y} D^2$$

   - And thus:

$$\lambda_k = \frac{d_k^2}{Y}$$

c. *Explain the theorem using layman's language. Use as few formulas as you can. You can draw a diagram if you think it is helpful. It is limited to 100-300 words. A figure may be counted as 50 words.*

Theorem 1.1 states that, in a space-time data matrix X, the relation between the locations / the spatial patterns can be found in the SVD results as well as in the eigenvectors of the covariance matrix.

It states that the eigenvector of the covariance matrix of X with the largest eigenvalue is the vector pointing in the direction where the data has the biggest variance. The eigenvector with the second largest eigenvalues points in the direction of the vector where the data has its second biggest variance, and so on.

It furthermore states that the eigenvalues of the covariance matrix are capture in the singular values of the SVD, where the eigenvalues are the square of the singular values divided by the number time dimensions of X.

4. **Concept of independent vectors**

a. Show that the three row-vectors of the following matrix A are not linearly independent

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 6 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Vectors are not linearly independent if they can be expressed as multiples of each other. With the three row vectors of A, we can show that $Row\ 3$ is a multiple of $Row\ 2$ and $Row\ 1$:

$$Row\ 3 = \frac{1}{4} \times Row\ 1\ -\ \frac{1}{8} \times Row\ 2$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} = \frac{1}{4} \times \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} - \frac{1}{8} \times \begin{bmatrix} 2 & 4 & 6 & 0 \end{bmatrix}$$

b. Columns 2, 3, and 4 of the above matrix A form a square matrix, denoted by H. Use R to compute the determinant of this matrix H. Does this matrix H have an inverse?

$$H = \begin{bmatrix} 2 & 3 & 4 \\ 4 & 6 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
H <- matrix(
    c(
        2, 3, 4,
        4, 6, 0,
        0, 0, 1
    ),
    ncol = 3
)

det_H <- det(H)
det_H
# [1] 6.661338e-16
# This is a very small number close to zero, can be considered zero
```

Since the determinant of H is so close to zero, it can be considered zero and thus it has no inverse. It is linearly dependant and reduces space to a lower dimension.

c. Use R to compute the inverse matrix of the following matrix B:

$$B = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 0 \\ 7 & 1 & 9 & 0 \\ 0 & 1 & 0 & 8 \end{bmatrix}$$

```
B <- matrix(
    c(
        1, 2, 3, 4,
        4, 5, 6, 0,
        7, 1, 9, 0,
        0, 1, 0, 8
    ),
    ncol = 4
)

inv_B <- solve(B)
inv_B
#            [,1]        [,2]        [,3]        [,4]
# [1,] -0.8666667 -0.13333333  0.68888889  0.01666667
# [2,]  0.2333333  0.26666667 -0.21111111 -0.03333333
# [3,]  0.1333333 -0.13333333  0.02222222  0.01666667
# [4,]  0.4333333  0.06666667 -0.34444444  0.11666667
```

d. Use R to compute the determinant of the matrix B.

```
det_B <- det(B)
det_B
# [1] -360
```

e. Use R to find all the four unit eigenvectors of B and their corresponding eigenvalues of B.

```
eig_B <- eigen(B)

eigenvalues_B <- eig_B$values
unit_eigenvectors_B <- eig_B$vectors
# the eigen() func already normalizes the eigenvectors to unit length

eigenvalues_B
# [1] 13.200401  7.269414  3.578543 -1.048358

unit_eigenvectors
# [,1]       [,2]       [,3]       [,4]
# [1,] -0.5027881  0.1720017  0.3888892 -0.8857564
# [2,] -0.2586260 -0.1561739 -0.6696376  0.2046320
```

```
# [3,] -0.7285304  0.2432922  0.5259025  0.1422598
# [4,] -0.3867302 -0.9417187 -0.3518199  0.3915656
```

f. Are the first two unit eigenvectors from Part (e)'s results orthogonal? You must use R to do some computing to substantiate your answer.

```
dot_product <- sum(unit_eigenvectors[, 1] * unit_eigenvectors[, 2])
dot_product
# [1] 0.1408555
# The unit eigenvectors are not orthogonal
# since the dot product between them is
# not close to zero
```