Homework #06

# 1a

i)

A=[-2, -4, 0; 4, -0, 0; 0, -8, 0; -2, -20, 0];

b=[-1; -3; -1; -3];

disp('attempt to solve with least sqs:')

attempt to solve with least sqs:

x=inv(A.'\*A)\*(A.'\*b)

Warning: Matrix is singular to working precision.

x = 3×1

NaN

NaN

NaN

We get NaNs.

This is because the system is formally overdetermined (the equations are contradictory), so the rank is less than N (where [M,N]=size(A)). So A^T\*A, an NxN matrix, is non-invertible.

disp('(A^T \* A)^-1:')

(A^T \* A)^-1:

disp(inv(A.'\*A))

Warning: Matrix is singular to working precision.

Inf Inf Inf

Inf Inf Inf

Inf Inf Inf

But this method requires inverting (A^T\*A) so we can't use it.

ii)

disp('A^T \* A:')

A^T \* A:

disp(A.'\*A)

24 48 0

48 480 0

0 0 0

disp('A^T \* b:')

A^T \* b:

disp(A.'\*b)

-4

72

0

one\_sol=pinv(A)\*b

one\_sol = 3×1

-0.5833

0.2083

0

second\_sol=[one\_sol(1,1); one\_sol(2,1); 99]

second\_sol = 3×1

-0.5833

0.2083

99.0000

1iii)

Assumptions: 1) Minimizes the error, but also 2) require that the solution’s norm is as small as possible. This is necessary to have 1 unique solution since A^T \* A \* x = A^T \* b already has infinite solutions (z can vary).

x\_unique=pinv(A)\*b

x\_unique = 3×1

-0.5833

0.2083

0

iv)

disp('from Matlab:')

from Matlab:

x\_matlab=A\b;

Warning: Rank deficient, rank = 2, tol = 1.945901e-14.

disp(x\_matlab)

-0.5833

0.2083

0

disp('norm of x from Matlab:')

norm of x from Matlab:

disp(norm(x\_matlab))

0.6194

disp('x calculated from pseudoinverse:')

x calculated from pseudoinverse:

disp(x\_unique)

-0.5833

0.2083

0

disp('norm of x from pseudoinverse:')

norm of x from pseudoinverse:

disp(norm(x\_unique))

0.6194

They are the same.

# 1b

A=[3, 6, 1;

6, 12, 2];

b=[-5;

-4];

i) The rank of A is 1 because the second equation is the first equation x 2; A is formally underdetermined and r<M.

An x that exactly solves this cannot be found because it has two contradictory equations (same equation from A and two distinct b entries).

ii) Assumptions: 1) Minimizes the error, but also 2) require that the solution’s norm is as small as possible. This is necessary to have 1 unique solution since A^T \* A \* x = A^T \* b already has infinite solutions:

disp('A^T \* A:')

A^T \* A:

disp(A.'\*A)

45 90 15

90 180 30

15 30 5

disp('A^T \* b:')

A^T \* b:

disp(A.'\*b)

-39

-78

-13

(all the same equation)

Calculation:

x\_unique=pinv(A)\*b

x\_unique = 3×1

-0.1696

-0.3391

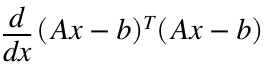
-0.0565

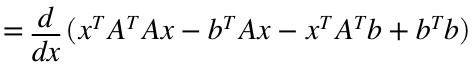
# 2

2a)

So for our least sqs solution, we want to minimize the distance between Ax and b,, which is = 

So we have an optimization problem and want to look for where the derivative = 0:







Therefore,  and 

A=[-4, -1, -1;

8, -10, 0;

2, 5, -2;

-7, 0, -7;

-7, 1, -3];

b=[-7; -3; -6; -2; -9];

x=inv(A.'\*A)\*(A.'\*b)

x = 3×1

0.0904

0.1189

0.8745

2b)

v1 = A(:,1);

v2 = A(:,2);

v3 = A(:,3);

y1\_hat=v1 / norm(v1);

y2=v2 - ((v2.'\*y1\_hat).\*y1\_hat);

y2\_hat = y2 / norm(y2);

y3=v3 - ((v3.'\*y1\_hat).\*y1\_hat) - ((v3.'\*y2\_hat).\*y2\_hat);

y3\_hat = y3 / norm(y3);

Q=[y1\_hat y2\_hat y3\_hat]

Q = 5×3

-0.2965 -0.2635 0.1672

0.5930 -0.6870 -0.3389

0.1482 0.5870 -0.6440

-0.5189 -0.2840 -0.6651

-0.5189 -0.1829 -0.0018

R=Q.'\*A

R = 3×3

13.4907 -5.4111 5.1887

-0.0000 9.8853 1.6263

0.0000 -0.0000 5.7820

2c)

% TF said I can use Matlab to directly multiply

x\_qr=(inv(R)\*Q.'\*b)

x\_qr = 3×1

0.0904

0.1189

0.8745

2d)

They are the same.

# 3

3a)

Both the SVD and Covariance Matrix method give the same answer because using SVD to give U, sigma, and V, we know that U will have as columns the eigenvectors of F\*F^T and be normalized ((1/N)\*F\*F^T). (1/N)\*F\*F^T is the definition of the covariance matrix, whose eigenvectors will be the PCs.

3b)

F=[-0.8, -14, 2.4, -8.4, 4, -2.4, 6, 4, 8.8;

-7.2, 22, -8.8, 15, -8, 6.4, -8, -2, -8.8;

26, 3.2, 14, -2.4, 4, -7.2, -6, -14, -18];

[M, N] = size(F);

F\_centered = F;

F\_means = mean(F, 2);

F\_centered(1,:) = F\_centered(1,:) - F\_means(1,1);

F\_centered(2,:) = F\_centered(2,:) - F\_means(2,1);

F\_centered(3,:) = F\_centered(3,:) - F\_means(3,1);

% using SVD

[U\_svd, sigma, V] = svd(F\_centered);

disp('Principal Components from SVD:')

Principal Components from SVD:

for i = 1:1:length(U\_svd)

disp(U\_svd(:, i))

end

0.2300

-0.0611

-0.9713

-0.4828

0.8594

-0.1684

0.8450

0.5076

0.1682

disp('Expansion Matrix:')

Expansion Matrix:

T\_svd=sigma\*V.'

T\_svd = 3×9

-25.0258 -7.7016 -12.5368 -0.5467 -2.5050 6.0211 7.6677 ⋯

-10.2662 25.0405 -11.1653 17.2642 -9.5662 7.7851 -8.8477

0.0526 -0.1133 -0.0735 0.1237 0.0030 0.0211 0.0113

% using covariance matrix

C=(1/N).\*F\_centered \* F\_centered.';

[U\_cov, lambda]=eig(C);

disp('Principal Components from Covariance Matrix:')

Principal Components from Covariance Matrix:

for i = 1:1:length(U\_cov)

disp(U\_cov(:, i))

end

-0.8450

-0.5076

-0.1682

-0.4828

0.8594

-0.1684

-0.2300

0.0611

0.9713

disp('Expansion Matrix:')

Expansion Matrix:

T\_cov=U\_cov.' \* F\_centered

T\_cov = 3×9

-0.0526 0.1133 0.0735 -0.1237 -0.0030 -0.0211 -0.0113 ⋯

-10.2662 25.0405 -11.1653 17.2642 -9.5662 7.7851 -8.8477

25.0258 7.7016 12.5368 0.5467 2.5050 -6.0211 -7.6677

They are the same PCs (different directions but along same eigenvectors) for both methods. Likewise, same expansion matrices, although the order of the rows depends on the order of the PCs.

3c) Grades for different courses for different students:

From the largest eigenvalue's PC, student 1 and student 3 are somewhat oppositely correlated where student 3 is just really good at like maybe science courses, but student 1 is better at math courses.

But simultaneously from the second largest eigenvalue's PC maybe student 1 and 3 also talk to each other in class, so student 2 does the best when both student 1 and 3 aren't doing great and are distracting each other, but does worst when student 1 and 3 talking to each other helps each other out. Student 2 is oppositely correlated with both student 1 and 3.

3d)

load global\_temperature.mat

% lat 1x36, lon 1x72, T1 72x36x138, years 1x138

load global\_topography.mat

% topo 180x90, X 180x1, Y 90x1

figure; hold on; colormap(jet)

title('first year')

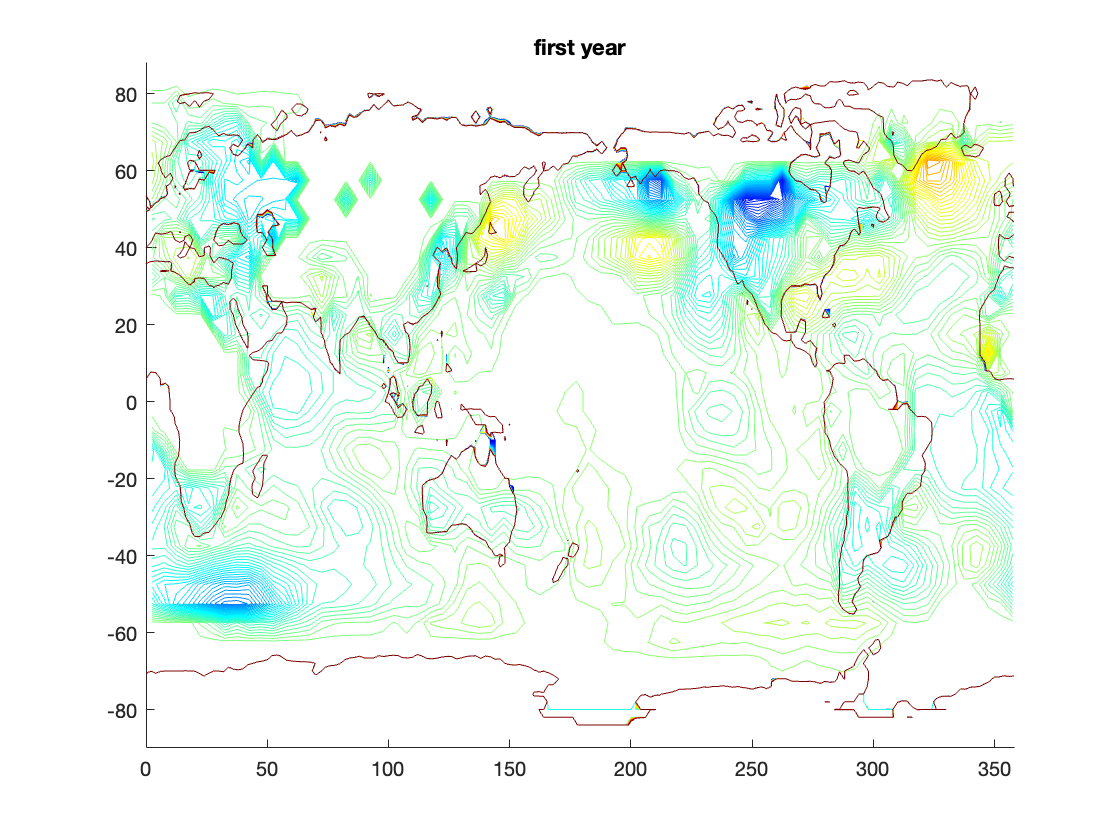
TMP=permute(squeeze(T1(:,:,1)),[2,1]);

contour(lon,lat,TMP,linspace(-5,5,100))

TOP=permute(squeeze(topo(:,:,1)),[2,1]);

contour(X,Y,TOP,linspace(-5,5,100))

hold off



figure; hold on; colormap(jet)

title('last year')

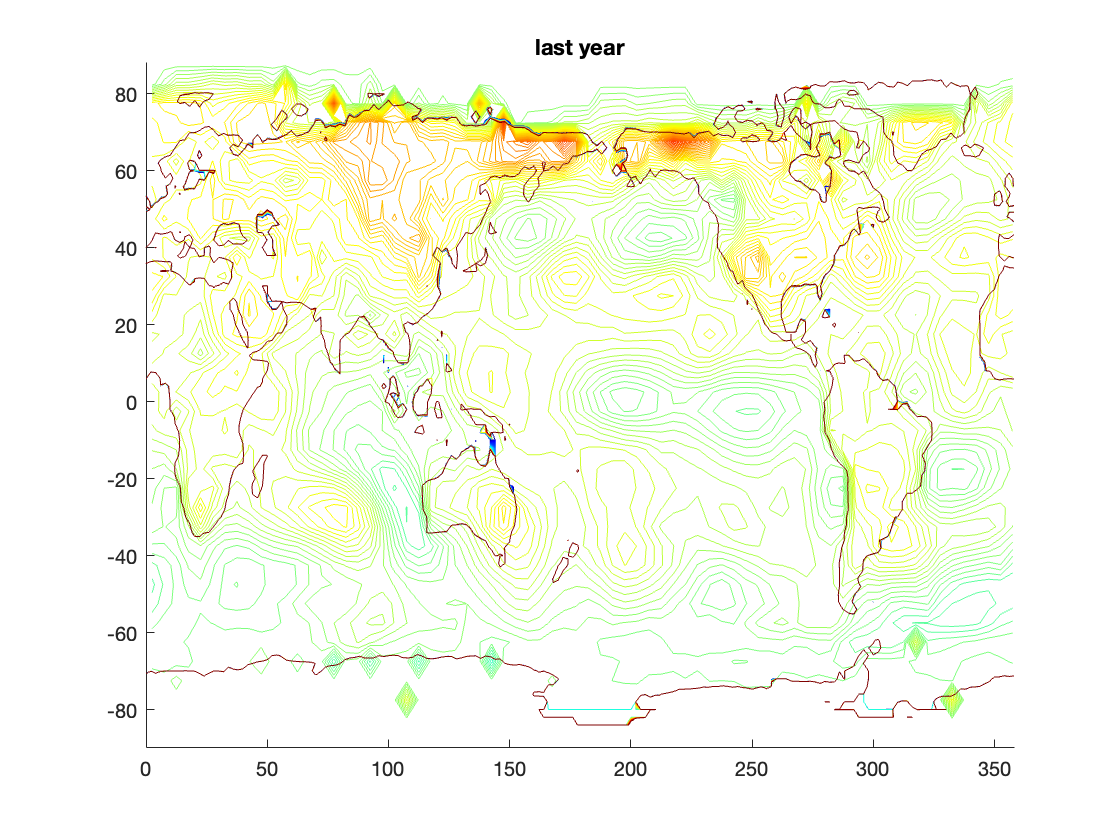
TMP=permute(squeeze(T1(:,:,end)),[2,1]);

contour(lon,lat,TMP,linspace(-5,5,100))

TOP=permute(squeeze(topo(:,:,end)),[2,1]);

contour(X,Y,TOP,linspace(-5,5,100))

hold off



[size1, size2, size3]=size(T1);

N=size1\*size2;

% reshape() will keep third dimension correctly

T1\_reshape = reshape(T1, [N, size3]);

% permute rows tho so PC time series goes up

[~,idx] = sort(T1\_reshape(:,1));

T1\_reshape = T1\_reshape(idx,:);

T2 = T1\_reshape;

T\_means = mean(T1\_reshape, 2);

for i=1:1:N

T2(i,:) = T2(i,:) - T\_means(i,1);

end

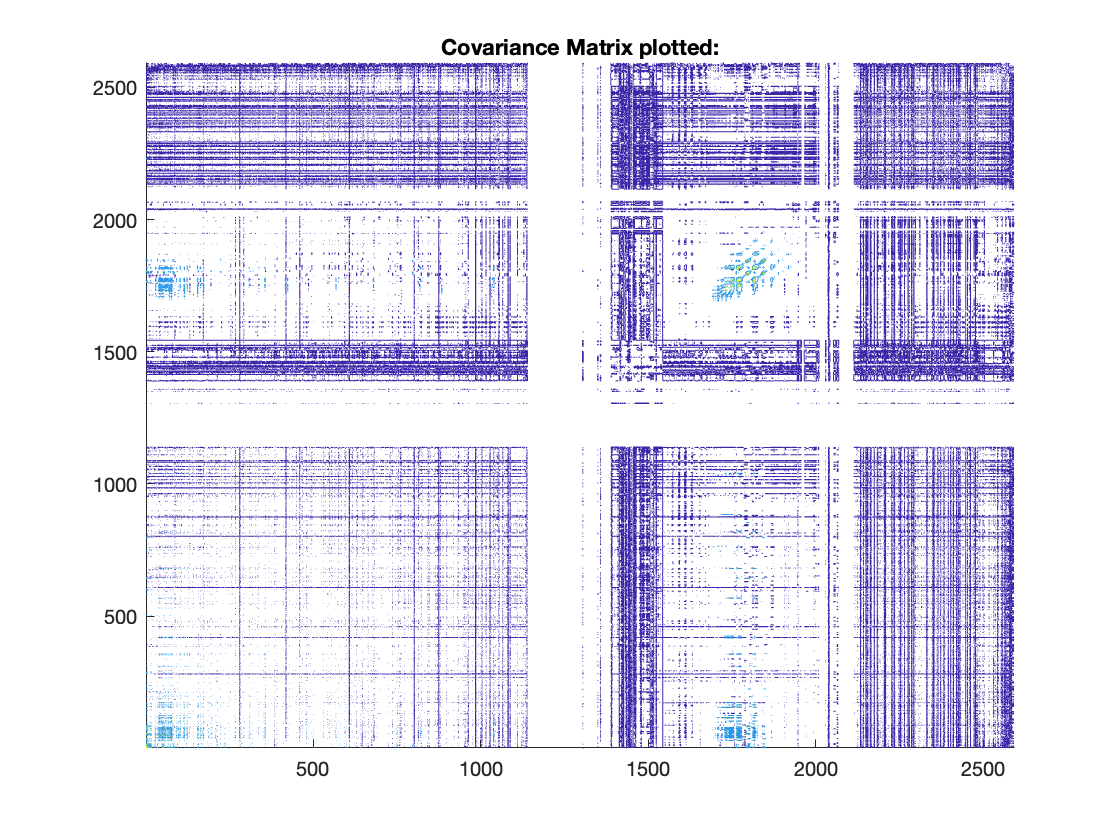
T\_cov = (1/size3).\* T2 \* T2.';

figure; hold on

title('Covariance Matrix plotted:')

contour(T\_cov)

hold off



Top 3 PCs:

[T\_U, T\_lambda]=eig(T\_cov);

% eig() will go from greatest lambda to smallest

pc1=T\_U(:, 1);

pc2=T\_U(:, 2);

pc3=T\_U(:, 3);

Time series for the top 3 PCs:

T\_T = T\_U.' \* T2;

figure; hold on;

title('Top 3 PCs time series')

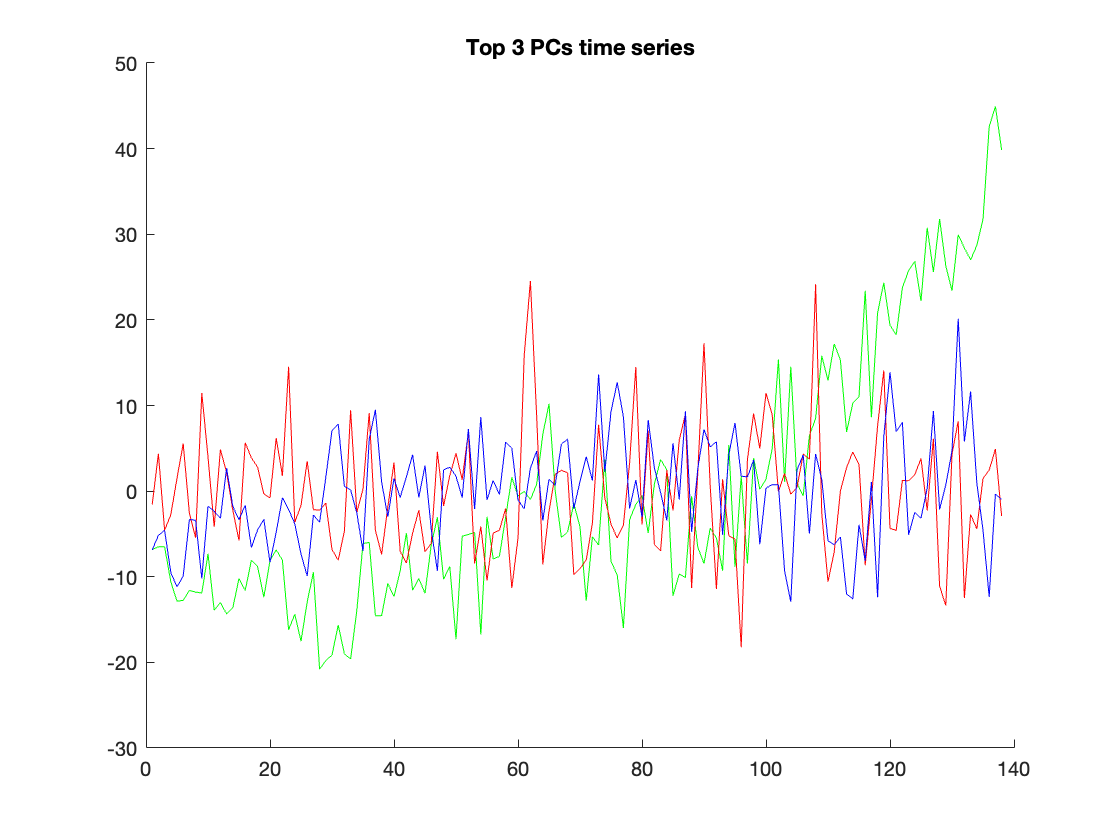
pc\_color = ['g', 'r', 'b'];

for i = 1:1:3

plot(1:size3, T\_T(i, :), pc\_color(1, i))

end

hold off



So the first PC that's going higher and higher is showing global temperature going higher and higher, and in fact overall, this PC explains the most variance. So that's global warming. And then the second and third PC's are likely hemisphere seasonal differences, where north and south are oppositely correlated which is why the second and third PC are oscillating in opposition to each other.

# 4

X=[ 0.1, 2.8, 3.5, 0.5, 1.2, 4, 2.1, 0, 2.9, 3.6, 0.4, 1.1;

3.8, 1.4, 0.5, 3.3, 2.7, 0, 1.7, 4.1, 1.1, 0.5, 3.5, 2.9;

9.7, -1.3, -4.2, 8.1, 5.3, -5.8, 1.8, 9.9, -1.5, -4.2, 8.2, 5.5;

9.9, -1.7, -4.3, 8.5, 5.7, -5.8, 2.3, 9.7, -1.5, -4.3, 8.3, 5.5];

Y=[-7.6, -3.6, -2.8, -6.9, -6.1, -2.2, -4.9, -7.4, -3.9, -2.7, -7, -6.1;

-2.1, -6.5, -7.4, -2.9, -3.8, -8.1, -5.1, -2.4, -6.3, -7.5, -2.8, -3.8;

-2.7, -6, -6.9, -3.1, -4, -7.3, -5, -2.7, -6, -6.9, -3.1, -4];

4a)

X\_centered = X;

X\_means = mean(X, 2);

for i = 1:1:length(X\_means)

X\_centered(i,:) = X\_centered(i,:) - X\_means(i,1);

end

X\_normed = X\_centered ./ std(X\_centered(:));

Y\_centered = Y;

Y\_means = mean(Y, 2);

for i = 1:1:length(Y\_means)

Y\_centered(i,:) = Y\_centered(i,:) - Y\_means(i,1);

end

Y\_normed = Y\_centered ./ std(Y\_centered(:));

4b)

F=[X\_normed;Y\_normed]

F = 7×12

-0.4207 0.2284 0.3967 -0.3246 -0.1563 0.5169 0.0601 ⋯

0.4027 -0.1743 -0.3907 0.2825 0.1382 -0.5109 -0.1022

1.7010 -0.9436 -1.6409 1.3163 0.6431 -2.0255 -0.1983

1.7330 -1.0558 -1.6809 1.3964 0.7233 -2.0416 -0.0942

-1.2963 0.7778 1.1926 -0.9333 -0.5185 1.5037 0.1037

1.4475 -0.8340 -1.3006 1.0327 0.5660 -1.6636 -0.1080

1.0932 -0.6179 -1.0846 0.8858 0.4191 -1.2920 -0.0994

[U, sigma, V\_T]=svd(F)

U = 7×7

0.1297 -0.2030 -0.1036 0.0488 -0.0956 0.9328 -0.2229

-0.1251 0.5329 0.3742 -0.1554 -0.1040 -0.0008 -0.7248

-0.5154 0.5446 0.0344 -0.1519 -0.0716 0.3260 0.5496

-0.5245 -0.2051 -0.6543 -0.3434 -0.1862 -0.1166 -0.2976

0.3780 0.3970 -0.4529 -0.2838 0.6398 0.0549 -0.0383

-0.4158 -0.3964 0.4057 -0.2420 0.6610 0.0843 -0.0533

-0.3333 0.1463 -0.2241 0.8327 0.3065 -0.0003 -0.1731

sigma = 7×12

9.0511 0 0 0 0 0 0 ⋯

0 0.2204 0 0 0 0 0

0 0 0.1521 0 0 0 0

0 0 0 0.0584 0 0 0

0 0 0 0 0.0380 0 0

0 0 0 0 0 0.0209 0

0 0 0 0 0 0 0.0151

V\_T = 12×12

-0.3698 -0.2604 0.3177 -0.1512 0.4297 0.4848 0.2735 ⋯

0.2141 0.5092 0.1138 0.1845 0.4602 0.0622 -0.4823

0.3514 -0.0336 0.2055 -0.3449 0.1087 -0.5918 0.3104

-0.2835 -0.0149 -0.5641 0.2267 0.1138 -0.0571 0.1799

-0.1458 -0.2792 -0.0828 -0.2744 -0.2449 -0.0699 -0.6667

0.4349 0.0254 -0.2967 0.2315 -0.1209 0.1477 0.2842

0.0320 -0.3899 -0.3826 -0.0814 0.0292 -0.0437 -0.0880

-0.3596 0.5663 0.0056 -0.0650 -0.5225 0.0721 0.1507

0.2042 -0.3176 0.4420 0.5648 -0.3635 0.1292 -0.0450

0.3563 0.1309 -0.1034 -0.3619 0.0193 0.4095 0.0072

⋮

for i = 1:1:5

sigma\_subset=sigma(1:i, 1:i);

disp(['using ' num2str(i) ' PCs:'])

percent\_explained\_var=100\*sum(sigma\_subset(:))/sum(sigma(:))

end

using 1 PCs:

percent\_explained\_var = 94.7162

using 2 PCs:

percent\_explained\_var = 97.0225

using 3 PCs:

percent\_explained\_var = 98.6146

using 4 PCs:

percent\_explained\_var = 99.2261

using 5 PCs:

percent\_explained\_var = 99.6235

So just the first PC is enough to explain more than 50% of the total variance.

disp('First PC:')

First PC:

disp(U(:, 1))

0.1297

-0.1251

-0.5154

-0.5245

0.3780

-0.4158

-0.3333

So from this one PC, it looks like the first row of X (4 total rows) is correlated with the first row of Y (the bottom 3 rows) and then the rest of the rows of X and Y are correlated together in the opposite direction.

4c)

X\_normed

X\_normed = 4×12

-0.4207 0.2284 0.3967 -0.3246 -0.1563 0.5169 0.0601 ⋯

0.4027 -0.1743 -0.3907 0.2825 0.1382 -0.5109 -0.1022

1.7010 -0.9436 -1.6409 1.3163 0.6431 -2.0255 -0.1983

1.7330 -1.0558 -1.6809 1.3964 0.7233 -2.0416 -0.0942

Y\_normed.'

ans = 12×3

-1.2963 1.4475 1.0932

0.7778 -0.8340 -0.6179

1.1926 -1.3006 -1.0846

-0.9333 1.0327 0.8858

-0.5185 0.5660 0.4191

1.5037 -1.6636 -1.2920

0.1037 -0.1080 -0.0994

-1.1926 1.2920 1.0932

0.6222 -0.7302 -0.6179

1.2444 -1.3525 -1.0846

⋮

Diagram

Description automatically generated with low confidence

[M,N]=size(X\_normed);

C=X\_normed\*Y\_normed.'\*(1/N)

C = 4×3

0.3343 -0.3678 -0.2951

-0.3221 0.3544 0.2847

-1.3290 1.4622 1.1730

-1.3531 1.4887 1.1935

The covariance matrix is the variability between the two datasets, X and Y. So, C(1,1) would be the covariance of the 1st component for X and the 1st Y component.

4d)

[U, sigma, V\_T]=svd(C)

U = 4×4

-0.1712 0.1486 0.6814 0.6959

0.1650 -0.6093 -0.4619 0.6230

0.6807 -0.4570 0.5186 -0.2427

0.6929 0.6308 -0.2310 0.2620

sigma = 4×3

3.3758 0 0

0 0.0006 0

0 0 0.0000

0 0 0

V\_T = 3×3

-0.5784 -0.3424 -0.7404

0.6364 0.3784 -0.6722

0.5104 -0.8600 -0.0010

for i = 1:1:3

sigma\_subset=sigma(1:i, 1:i);

disp(['using ' num2str(i) ' PCs:'])

percent\_explained\_var=100\*sum(sigma\_subset(:))/sum(sigma(:))

end

using 1 PCs:

percent\_explained\_var = 99.9826

using 2 PCs:

percent\_explained\_var = 99.9998

using 3 PCs:

percent\_explained\_var = 100

Just the first singular value is able to explain 99% of the variance.

The U vectors represent the components in the X dataset. The V vectors represent the components in the Y dataset.

4e)

C\_squared = C.^2;

total\_covariance = sum(C\_squared(:))

total\_covariance = 11.3962

sigma\_squared = sigma.^2;

sigma\_squared = 4×3

11.3962 0 0

0 0.0000 0

0 0 0.0000

0 0 0

sum\_singular\_values = sum(sigma\_squared(:))

sum\_singular\_values = 11.3962

They are in fact equal.

Because the total covariance is the sum of the squared singular values, then the covariance contribution by one mode is the fraction of that one's squared singular value over the total sum of squared singular values used.

sigma\_squared

using 1 PCs:

percent\_explained\_var = 100.0000

using 2 PCs:

percent\_explained\_var = 100.0000

using 3 PCs:

percent\_explained\_var = 100

So since sigma\_squared is all basically zeros except the first singular value, we only need 1 mode to explain more than 50% of the covariance.

Total variance is the spread of the dataset while total covariance is the correlational directional between two varying data.