Table of Contents


```
A = [[0.3536 0 0.25 0.25];[0 -1.4142 -1 -1];[0.6124 0 0.433 -0.433]];
b = [1;1;1];

A_squared = A'*A
[V, S] = eig(A_squared);

% have to flip these because the MATLAB eig function returns them in an
% unexpected order
S_squared = fliplr(flipud(S))
```

```
A_squared =

    0.5001         0    0.3536   -0.1768
         0    2.0000    1.4142    1.4142
    0.3536    1.4142    1.2500    0.8750
   -0.1768    1.4142    0.8750    1.2500


S_squared =

    4.0657         0         0         0
         0    0.8221         0         0
         0         0    0.1122         0
         0         0         0   -0.0000
```

1.a

non-zero eigenvectors (orthonormal)

```
V1 = fliplr(V(:,2:4))
```

```
V1 =

    0.0255   -0.7631   -0.4088
    0.6953    0.0774    0.5104
    0.5097   -0.4848    0.0719
```

```
    0.5060    0.4203   -0.7532
```

1.b

zero eigenvectors

```
V2 = fliplr(V(:,1))
```

```
V2 =

    0.5000
    0.5000
   -0.7071
   -0.0000
```

1.c

```
A_squared_2 = [V1 V2] * S_squared * [V1'; V2']
```

```
A_squared_2 =

    0.5001    0.0000    0.3536   -0.1768
    0.0000    2.0000    1.4142    1.4142
    0.3536    1.4142    1.2500    0.8750
   -0.1768    1.4142    0.8750    1.2500
```

1.d

we only have two (nonzero) singular values here, so we select them for S

```
S_tilde = sqrt(S_squared(1:3,1:3))

% calculate left singular vectors - using the non-zero singular values
U1 = A*V1*S_tilde
U1_tilde = [U1(:,1)/norm(U1(:,1)) U1(:,2)/norm(U1(:,2)) U1(:,3)/
norm(U1(:,3))]

% if this is zero, then U1 is orthogonal
dot(U1(:,1),(U1(:,2)))
```

```
S_tilde =

    2.0164         0         0
         0    0.9067         0
         0         0    0.3350
```

```
U1 =

    0.5302   -0.2593   -0.1055
   -4.0309   -0.0408   -0.0136
    0.0346   -0.7791    0.0358


U1_tilde =

    0.1304   -0.3154   -0.9400
   -0.9914   -0.0496   -0.1209
    0.0085   -0.9477    0.3191


ans =

   4.7184e-16
```

1.e

selected by inspection

```
U2 = [0; 0 ;0];
U2_tilde = [U2(:,1)/norm(U2(:,1))]
```

```
U2_tilde =

   NaN
   NaN
   NaN
```

1.f - Actually solve Ax=b

```
V1_tilde = V1;

% calculate using the SVD
A_dagger = V1_tilde * inv(S_tilde) * U1_tilde';

% solve for least square x
x_tilde = A_dagger*b
```

```
x_tilde =

    1.9992
   -1.5361
    0.3272
    0.8452
```

Published with MATLAB® R2023b