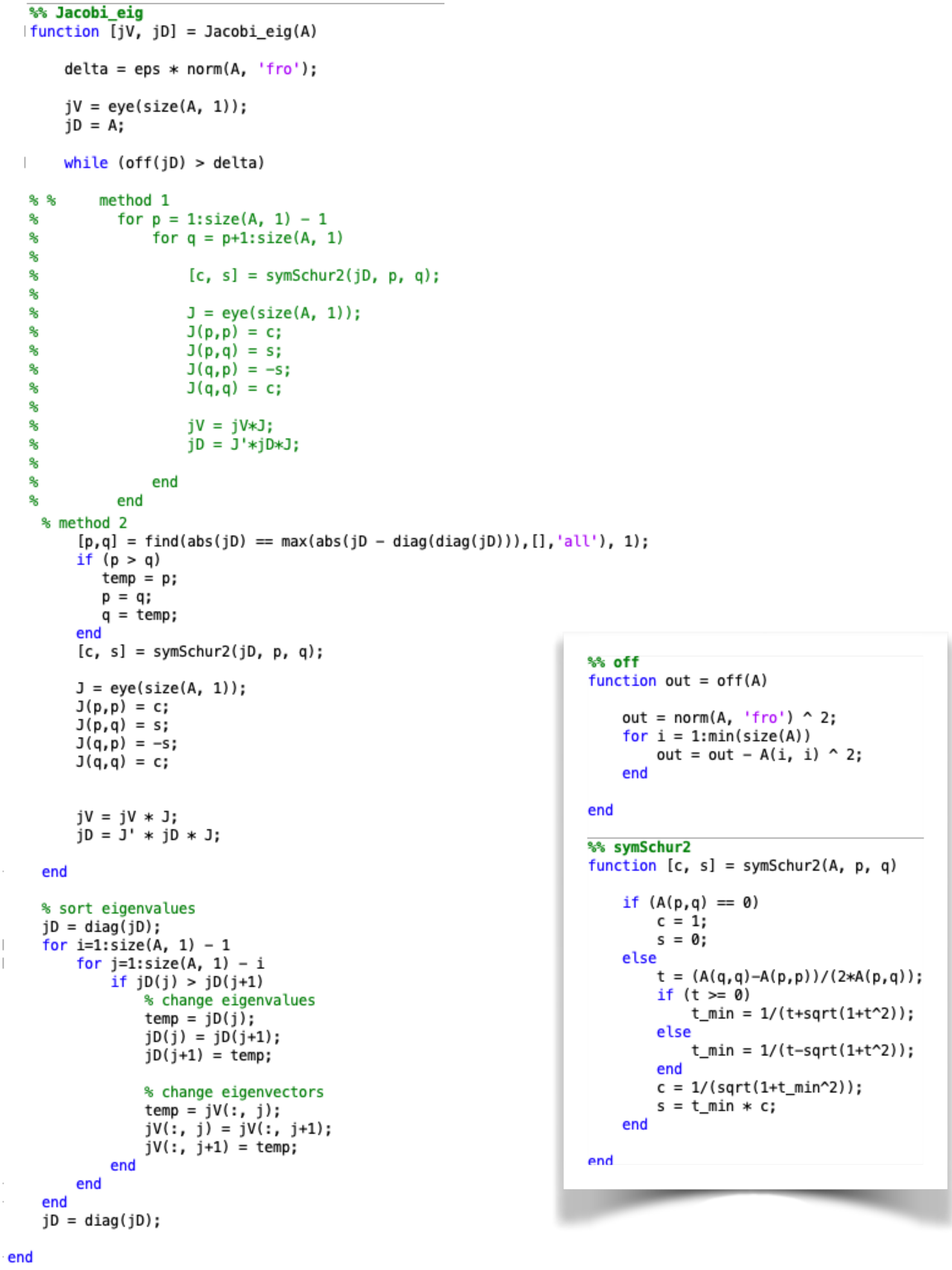
.۱

در این بخش از دو روش حلقه ای و پیدا کردن بزرگ اندازه ترین عنصر خارج قطر اصلی استفاده کردم در ادامه کد و خروجی آنها آورده شده است.

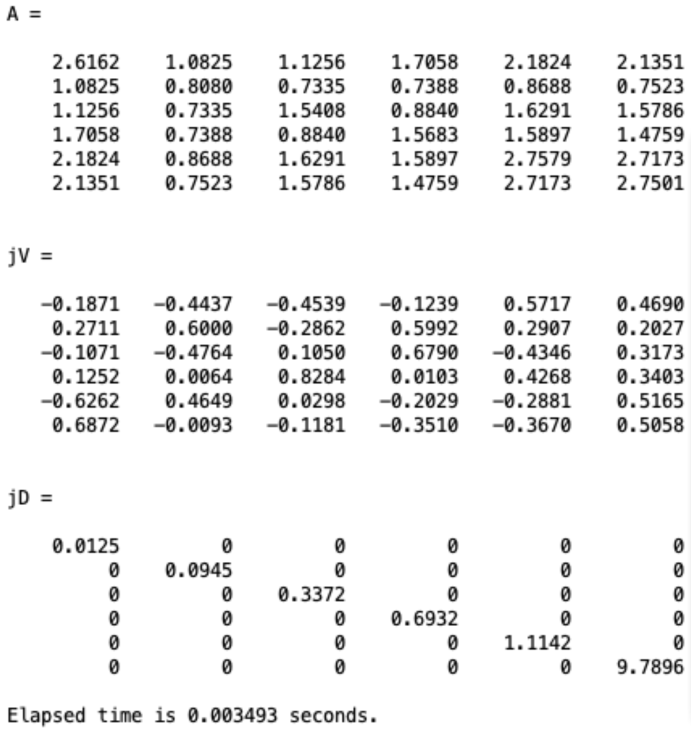


```
%% Jacobi_eig
function [jV, jD] = Jacobi_eig(A)

    delta = eps * norm(A, 'fro');

    jV = eye(size(A, 1));
    jD = A;

    while (off(jD) > delta)

% %     method 1
%         for p = 1:size(A, 1) - 1
%             for q = p+1:size(A, 1)
%
%                 [c, s] = symSchur2(jD, p, q);
%
%                 J = eye(size(A, 1));
%                 J(p,p) = c;
%                 J(p,q) = s;
%                 J(q,p) = -s;
%                 J(q,q) = c;
%
%                 jV = jV*J;
%                 jD = J'*jD*J;
%
%             end
%         end
    % method 2
        [p,q] = find(abs(jD) == max(abs(jD - diag(diag(jD))),[],'all'), 1);
        if (p > q)
            temp = p;
            p = q;
            q = temp;
        end
        [c, s] = symSchur2(jD, p, q);

        J = eye(size(A, 1));
        J(p,p) = c;
        J(p,q) = s;
        J(q,p) = -s;
        J(q,q) = c;


        jV = jV * J;
        jD = J' * jD * J;

    end

    % sort eigenvalues
    jD = diag(jD);
    for i=1:size(A, 1) - 1
        for j=1:size(A, 1) - i
            if jD(j) > jD(j+1)
                % change eigenvalues
                temp = jD(j);
                jD(j) = jD(j+1);
                jD(j+1) = temp;

                % change eigenvectors
                temp = jV(:, j);
                jV(:, j) = jV(:, j+1);
                jV(:, j+1) = temp;
            end
        end
    end
    jD = diag(jD);

end
```

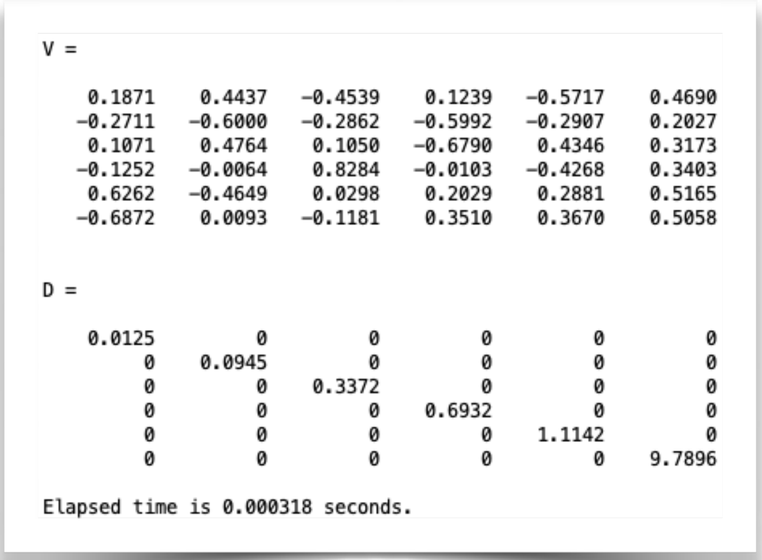
```
%% off
function out = off(A)

    out = norm(A, 'fro') ^ 2;
    for i = 1:min(size(A))
        out = out - A(i, i) ^ 2;
    end

end
```

```
%% symSchur2
function [c, s] = symSchur2(A, p, q)

    if (A(p,q) == 0)
        c = 1;
        s = 0;
    else
        t = (A(q,q)-A(p,p))/(2*A(p,q));
        if (t >= 0)
            t_min = 1/(t+sqrt(1+t^2));
        else
            t_min = 1/(t-sqrt(1+t^2));
        end
        c = 1/(sqrt(1+t_min^2));
        s = t_min * c;
    end

end
```

خروجی روش بزرگترین یاب برای ماتریس رندوم متقارن ۶ در ۶ و مقایسه با دستور eig (مقدار ضریب دلتا eps در نظر گرفته شد)

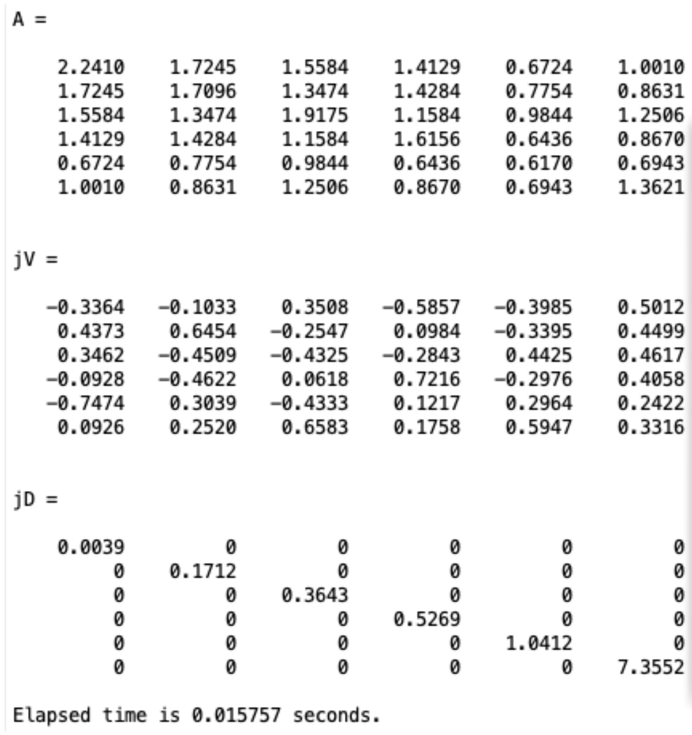


```
A =

    2.6162    1.0825    1.1256    1.7058    2.1824    2.1351
    1.0825    0.8080    0.7335    0.7388    0.8688    0.7523
    1.1256    0.7335    1.5408    0.8840    1.6291    1.5786
    1.7058    0.7388    0.8840    1.5683    1.5897    1.4759
    2.1824    0.8688    1.6291    1.5897    2.7579    2.7173
    2.1351    0.7523    1.5786    1.4759    2.7173    2.7501


jV =

   -0.1871   -0.4437   -0.4539   -0.1239    0.5717    0.4690
    0.2711    0.6000   -0.2862    0.5992    0.2907    0.2027
   -0.1071   -0.4764    0.1050    0.6790   -0.4346    0.3173
    0.1252    0.0064    0.8284    0.0103    0.4268    0.3403
   -0.6262    0.4649    0.0298   -0.2029   -0.2881    0.5165
    0.6872   -0.0093   -0.1181   -0.3510   -0.3670    0.5058


jD =

    0.0125         0         0         0         0         0
         0    0.0945         0         0         0         0
         0         0    0.3372         0         0         0
         0         0         0    0.6932         0         0
         0         0         0         0    1.1142         0
         0         0         0         0         0    9.7896

Elapsed time is 0.003493 seconds.
```

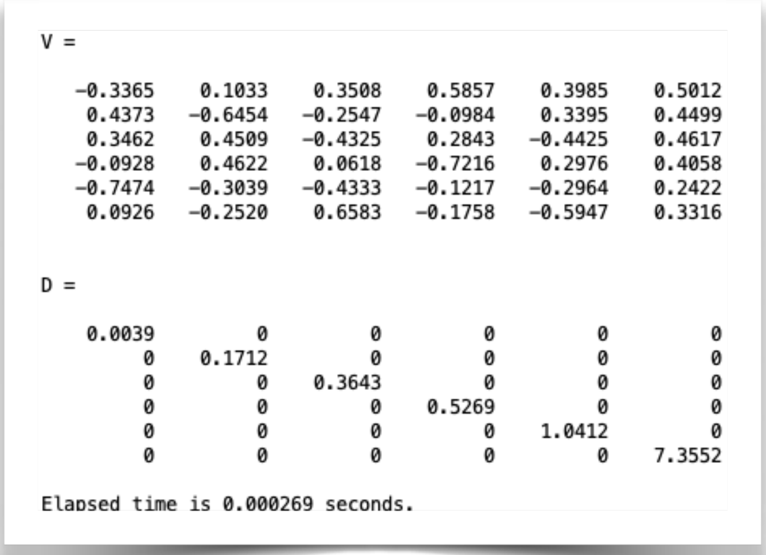


```
V =

    0.1871    0.4437   -0.4539    0.1239   -0.5717    0.4690
   -0.2711   -0.6000   -0.2862   -0.5992   -0.2907    0.2027
    0.1071    0.4764    0.1050   -0.6790    0.4346    0.3173
   -0.1252   -0.0064    0.8284   -0.0103   -0.4268    0.3403
    0.6262   -0.4649    0.0298    0.2029    0.2881    0.5165
   -0.6872    0.0093   -0.1181    0.3510    0.3670    0.5058


D =

    0.0125         0         0         0         0         0
         0    0.0945         0         0         0         0
         0         0    0.3372         0         0         0
         0         0         0    0.6932         0         0
         0         0         0         0    1.1142         0
         0         0         0         0         0    9.7896

Elapsed time is 0.000318 seconds.
```

خروجی روش حلقه ای برای ماتریس رندوم متقارن ۶ در ۶ و مقایسه با دستور eig (مقدار ضریب دلتا اول eps در نظر گرفته شد و پایان نیافت و در نتیجه یک مقدار بزرگتر ده به توان منفی ۴ در نظر گرفته شد. چون روش حلقوی اجرا میشود معمولا پاسخ خوب به مقدار اصلی دستور eig نزدیک میشود اما چون ابعاد ماتریس کوچک است و جستجوی بزرگترین عنصر خارج قطر اصلی خیلی هزینه ندارد، این روش کندتر از روش قبل اجرا میشود)

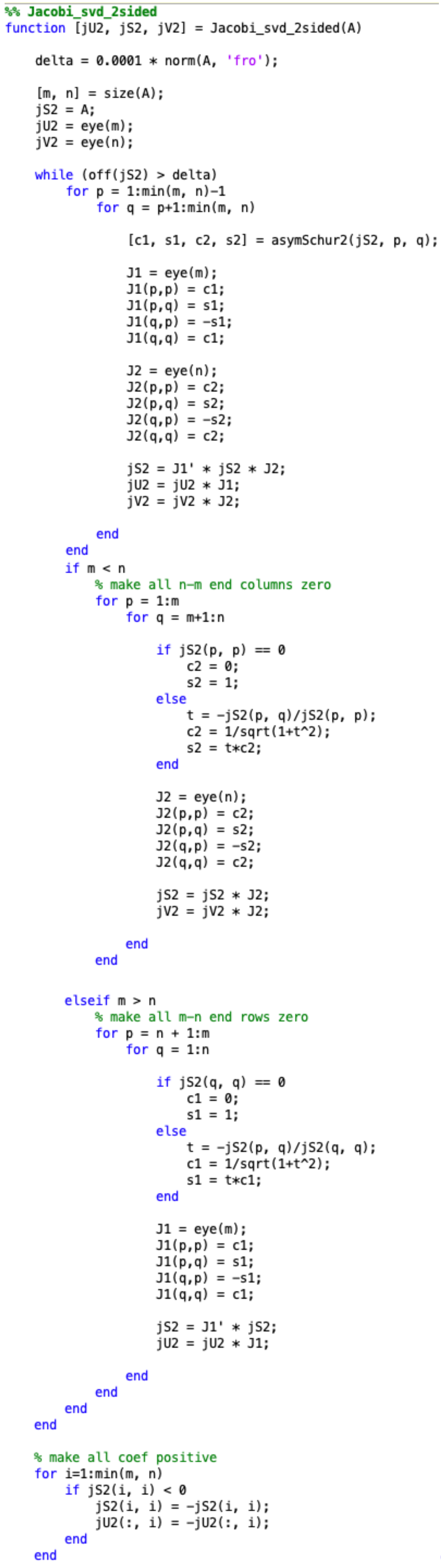


```
A =

    2.2410    1.7245    1.5584    1.4129    0.6724    1.0010
    1.7245    1.7096    1.3474    1.4284    0.7754    0.8631
    1.5584    1.3474    1.9175    1.1584    0.9844    1.2506
    1.4129    1.4284    1.1584    1.6156    0.6436    0.8670
    0.6724    0.7754    0.9844    0.6436    0.6170    0.6943
    1.0010    0.8631    1.2506    0.8670    0.6943    1.3621


jV =

   -0.3364   -0.1033    0.3508   -0.5857   -0.3985    0.5012
    0.4373    0.6454   -0.2547    0.0984   -0.3395    0.4499
    0.3462   -0.4509   -0.4325   -0.2843    0.4425    0.4617
   -0.0928   -0.4622    0.0618    0.7216   -0.2976    0.4058
   -0.7474    0.3039   -0.4333    0.1217    0.2964    0.2422
    0.0926    0.2520    0.6583    0.1758    0.5947    0.3316


jD =

    0.0039         0         0         0         0         0
         0    0.1712         0         0         0         0
         0         0    0.3643         0         0         0
         0         0         0    0.5269         0         0
         0         0         0         0    1.0412         0
         0         0         0         0         0    7.3552

Elapsed time is 0.015757 seconds.
```

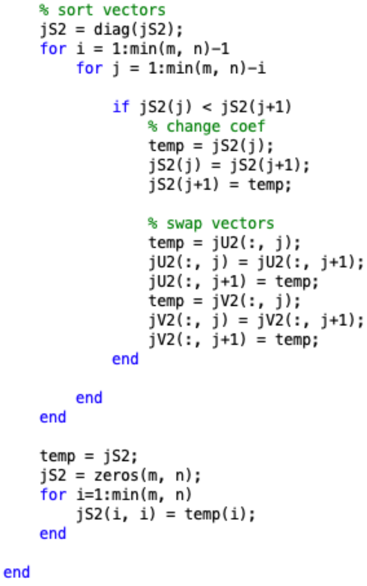


```
V =

   -0.3365    0.1033    0.3508    0.5857    0.3985    0.5012
    0.4373   -0.6454   -0.2547   -0.0984    0.3395    0.4499
    0.3462    0.4509   -0.4325    0.2843   -0.4425    0.4617
   -0.0928    0.4622    0.0618   -0.7216    0.2976    0.4058
   -0.7474   -0.3039   -0.4333   -0.1217   -0.2964    0.2422
    0.0926   -0.2520    0.6583   -0.1758   -0.5947    0.3316


D =

    0.0039         0         0         0         0         0
         0    0.1712         0         0         0         0
         0         0    0.3643         0         0         0
         0         0         0    0.5269         0         0
         0         0         0         0    1.0412         0
         0         0         0         0         0    7.3552

Elapsed time is 0.000269 seconds.
```

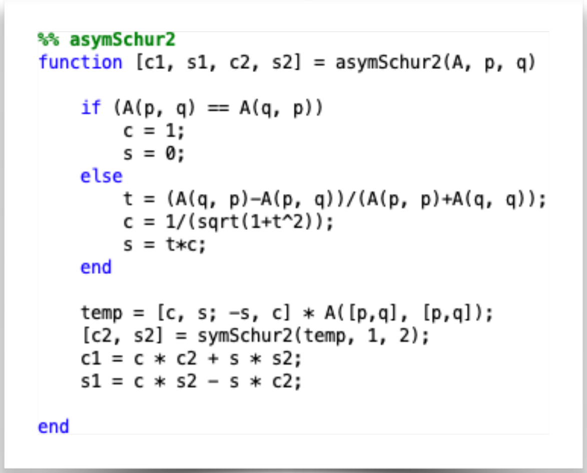
۲.



```
%% Jacobi_svd_2sided
function [jU2, jS2, jV2] = Jacobi_svd_2sided(A)

    delta = 0.0001 * norm(A, 'fro');

    [m, n] = size(A);
    jS2 = A;
    jU2 = eye(m);
    jV2 = eye(n);

    while (off(jS2) > delta)
        for p = 1:min(m, n)-1
            for q = p+1:min(m, n)

                [c1, s1, c2, s2] = asymSchur2(jS2, p, q);

                J1 = eye(m);
                J1(p,p) = c1;
                J1(p,q) = s1;
                J1(q,p) = -s1;
                J1(q,q) = c1;

                J2 = eye(n);
                J2(p,p) = c2;
                J2(p,q) = s2;
                J2(q,p) = -s2;
                J2(q,q) = c2;

                jS2 = J1' * jS2 * J2;
                jU2 = jU2 * J1;
                jV2 = jV2 * J2;

            end
        end
        if m < n
            % make all n-m end columns zero
            for p = 1:m
                for q = m+1:n

                    if jS2(p, p) == 0
                        c2 = 0;
                        s2 = 1;
                    else
                        t = -jS2(p, q)/jS2(p, p);
                        c2 = 1/sqrt(1+t^2);
                        s2 = t*c2;
                    end

                    J2 = eye(n);
                    J2(p,p) = c2;
                    J2(p,q) = s2;
                    J2(q,p) = -s2;
                    J2(q,q) = c2;

                    jS2 = jS2 * J2;
                    jV2 = jV2 * J2;

                end
            end


        elseif m > n
            % make all m-n end rows zero
            for p = n + 1:m
                for q = 1:n

                    if jS2(q, q) == 0
                        c1 = 0;
                        s1 = 1;
                    else
                        t = -jS2(p, q)/jS2(q, q);
                        c1 = 1/sqrt(1+t^2);
                        s1 = t*c1;
                    end

                    J1 = eye(m);
                    J1(p,p) = c1;
                    J1(p,q) = s1;
                    J1(q,p) = -s1;
                    J1(q,q) = c1;

                    jS2 = J1' * jS2;
                    jU2 = jU2 * J1;

                end
            end
        end
    end

    % make all coef positive
    for i=1:min(m, n)
        if jS2(i, i) < 0
            jS2(i, i) = -jS2(i, i);
            jU2(:, i) = -jU2(:, i);
        end
    end
```

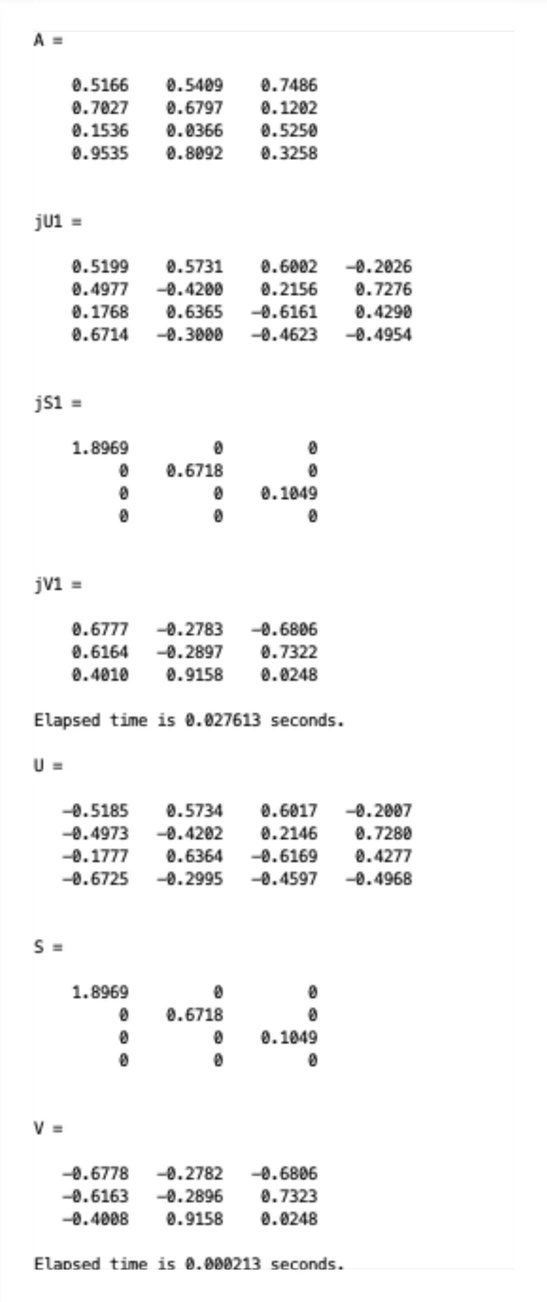


```
    % sort vectors
    jS2 = diag(jS2);
    for i = 1:min(m, n)-1
        for j = 1:min(m, n)-i

            if jS2(j) < jS2(j+1)
                % change coef
                temp = jS2(j);
                jS2(j) = jS2(j+1);
                jS2(j+1) = temp;

                % swap vectors
                temp = jU2(:, j);
                jU2(:, j) = jU2(:, j+1);
                jU2(:, j+1) = temp;
                temp = jV2(:, j);
                jV2(:, j) = jV2(:, j+1);
                jV2(:, j+1) = temp;
            end

        end
    end

    temp = jS2;
    jS2 = zeros(m, n);
    for i=1:min(m, n)
        jS2(i, i) = temp(i);
    end

end
```

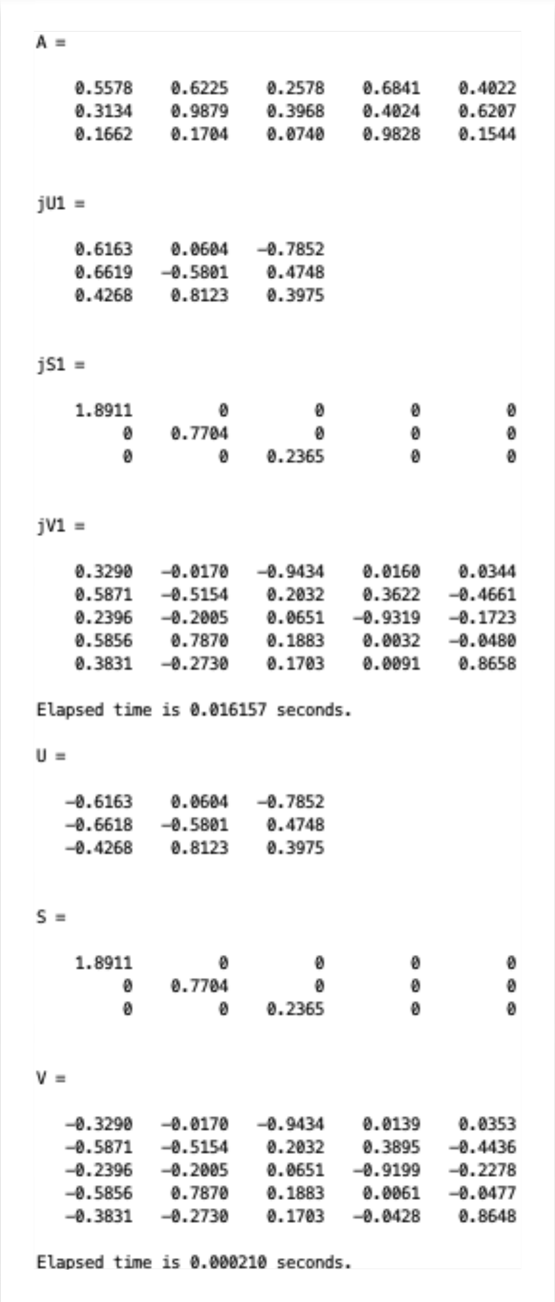


```
%% asymSchur2
function [c1, s1, c2, s2] = asymSchur2(A, p, q)

    if (A(p, q) == A(q, p))
        c = 1;
        s = 0;
    else
        t = (A(q, p)-A(p, q))/(A(p, p)+A(q, q));
        c = 1/(sqrt(1+t^2));
        s = t*c;
    end

    temp = [c, s; -s, c] * A([p,q], [p,q]);
    [c2, s2] = symSchur2(temp, 1, 2);
    c1 = c * c2 + s * s2;
    s1 = c * s2 - s * c2;

end
```

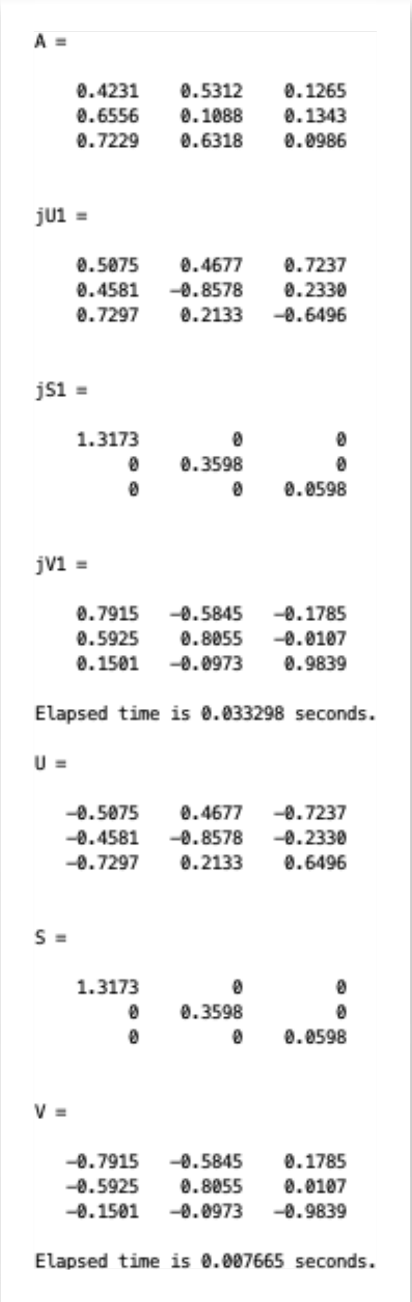
مقایسه روش با svd در ماتریس های رندوم با ابعاد مختلف :



```
A =

    0.5166    0.5409    0.7486
    0.7027    0.6797    0.1202
    0.1536    0.0366    0.5250
    0.9535    0.8092    0.3258


jU1 =

    0.5199    0.5731    0.6002   -0.2026
    0.4977   -0.4200    0.2156    0.7276
    0.1768    0.6365   -0.6161    0.4290
    0.6714   -0.3000   -0.4623   -0.4954


jS1 =

    1.8969         0         0
         0    0.6718         0
         0         0    0.1049
         0         0         0


jV1 =

    0.6777   -0.2783   -0.6806
    0.6164   -0.2897    0.7322
    0.4010    0.9158    0.0248

Elapsed time is 0.027613 seconds.

U =

   -0.5185    0.5734    0.6017   -0.2007
   -0.4973   -0.4202    0.2146    0.7280
   -0.1777    0.6364   -0.6169    0.4277
   -0.6725   -0.2995   -0.4597   -0.4968


S =

    1.8969         0         0
         0    0.6718         0
         0         0    0.1049
         0         0         0


V =

   -0.6778   -0.2782   -0.6806
   -0.6163   -0.2896    0.7323
   -0.4008    0.9158    0.0248

Elapsed time is 0.000213 seconds.
```



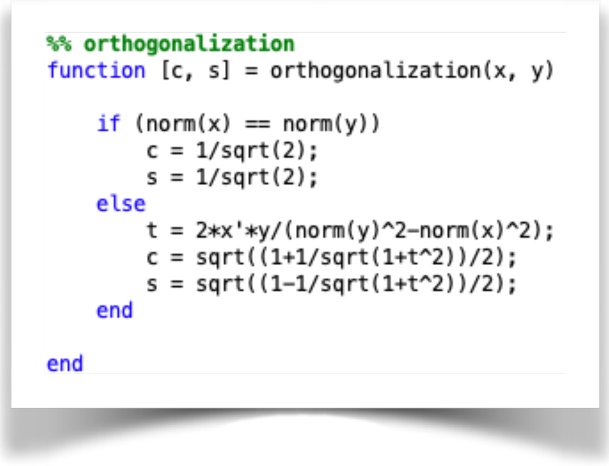
```
A =

    0.5578    0.6225    0.2578    0.6841    0.4022
    0.3134    0.9879    0.3968    0.4024    0.6207
    0.1662    0.1704    0.0740    0.9828    0.1544


jU1 =

    0.6163    0.0604   -0.7852
    0.6619   -0.5801    0.4748
    0.4268    0.8123    0.3975


jS1 =

    1.8911         0         0         0         0
         0    0.7704         0         0         0
         0         0    0.2365         0         0


jV1 =

    0.3290   -0.0170   -0.9434    0.0160    0.0344
    0.5871   -0.5154    0.2032    0.3622   -0.4661
    0.2396   -0.2005    0.0651   -0.9319   -0.1723
    0.5856    0.7870    0.1883    0.0032   -0.0480
    0.3831   -0.2730    0.1703    0.0091    0.8658

Elapsed time is 0.016157 seconds.

U =

   -0.6163    0.0604   -0.7852
   -0.6618   -0.5801    0.4748
   -0.4268    0.8123    0.3975


S =

    1.8911         0         0         0         0
         0    0.7704         0         0         0
         0         0    0.2365         0         0


V =

   -0.3290   -0.0170   -0.9434    0.0139    0.0353
   -0.5871   -0.5154    0.2032    0.3895   -0.4436
   -0.2396   -0.2005    0.0651   -0.9199   -0.2278
   -0.5856    0.7870    0.1883    0.0061   -0.0477
   -0.3831   -0.2730    0.1703   -0.0428    0.8648

Elapsed time is 0.000210 seconds.
```



```
A =

    0.4231    0.5312    0.1265
    0.6556    0.1088    0.1343
    0.7229    0.6318    0.0986


jU1 =

    0.5075    0.4677    0.7237
    0.4581   -0.8578    0.2330
    0.7297    0.2133   -0.6496


jS1 =

    1.3173         0         0
         0    0.3598         0
         0         0    0.0598


jV1 =

    0.7915   -0.5845   -0.1785
    0.5925    0.8055   -0.0107
    0.1501   -0.0973    0.9839

Elapsed time is 0.033298 seconds.

U =

   -0.5075    0.4677   -0.7237
   -0.4581   -0.8578   -0.2330
   -0.7297    0.2133    0.6496


S =

    1.3173         0         0
         0    0.3598         0
         0         0    0.0598


V =

   -0.7915   -0.5845    0.1785
   -0.5925    0.8055    0.0107
   -0.1501   -0.0973   -0.9839

Elapsed time is 0.007665 seconds.
```

۳.

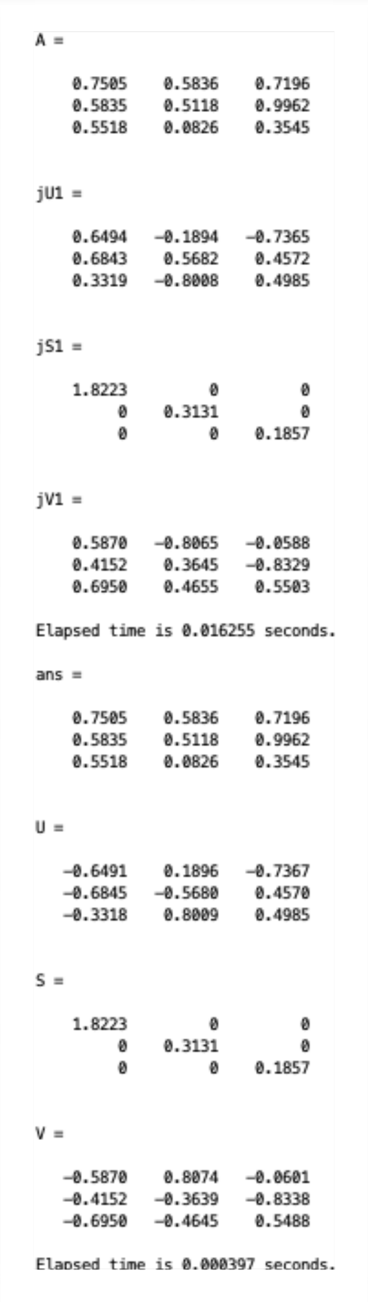
```
%% Jacobi_svd_1sided
function [jU1, jS1, jV1] = Jacobi_svd_1sided(A)

    [m, n] = size(A);

    if m <= n

        delta = 0.0001 * norm(A*A', 'fro');
        D = A';
        jV1 = eye(m);

        % Just work on m columns
        while (off(D'*D) > delta)
            for p = 1:m-1
                for q = p+1:m

                    [c, s] = orthogonalization(D(:, p), D(:, q));

                    J = eye(m);
                    J(p,p) = c;
                    J(p,q) = s;
                    J(q,p) = -s;
                    J(q,q) = c;

                    D = D * J;
                    jV1 = jV1 * J;

                end
            end
        end

        % get out JU1 & JS1 from D
        jS1 = zeros(n, m);
        jU1 = zeros(n, m);
        for i = 1:m
            jS1(i, i) = norm(D(:, i));
            jU1(:, i) = D(:, i)/norm(D(:, i));
        end

        % Transpose everything to get final result
        temp = jU1;
        jU1 = jV1;
        jS1 = jS1';
        jV1 = temp;

    else

    delta = 0.0001 * norm(A'*A, 'fro');
    D = A;
    jV1 = eye(n);

    % Just work on n columns
    while (off(D'*D) > delta)
        for p=1:n-1
            for q=p+1:n

                [c, s] = orthogonalization(D(:, p), D(:, q));

                J = eye(n);
                J(p,p) = c;
                J(p,q) = s;
                J(q,p) = -s;
                J(q,q) = c;

                D = D * J;
                jV1 = jV1 * J;

            end
        end
    end

    % get out JU1 & JS1 from D
    jS1 = zeros(m, n);
    jU1 = zeros(m, n);
    for i = 1:n
        jS1(i, i) = norm(D(:, i));
        jU1(:, i) = D(:, i)/norm(D(:, i));
    end

end
```



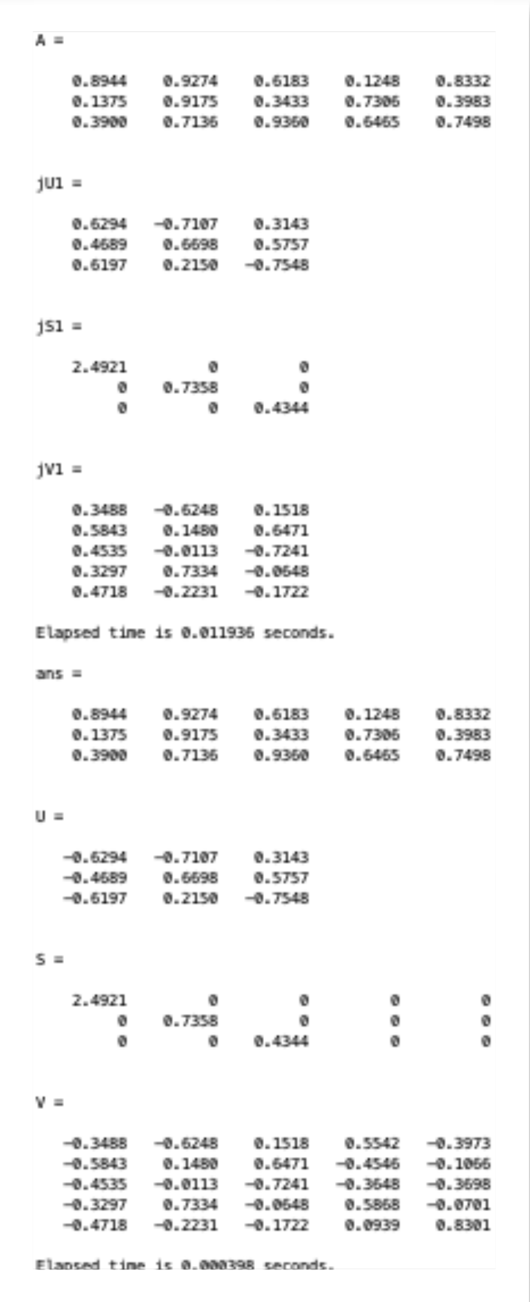
```
%% orthogonalization
function [c, s] = orthogonalization(x, y)

    if (norm(x) == norm(y))
        c = 1/sqrt(2);
        s = 1/sqrt(2);
    else
        t = 2*x'*y/(norm(y)^2-norm(x)^2);
        c = sqrt((1+1/sqrt(1+t^2))/2);
        s = sqrt((1-1/sqrt(1+t^2))/2);
    end

end
```

```
    jS1 = diag(jS1);

    for i = 1:min(m, n)-1
        for j = 1:min(m, n)-i

            if jS1(j) < jS1(j+1)

                % change coef
                temp = jS1(j);
                jS1(j) = jS1(j+1);
                jS1(j+1) = temp;

                % swap vectors
                temp = jU1(:, j);
                jU1(:, j) = jU1(:, j+1);
                jU1(:, j+1) = temp;
                temp = jV1(:, j);
                jV1(:, j) = jV1(:, j+1);
                jV1(:, j+1) = temp;

            end

        end
    end

    jS1 = diag(jS1);

end
```

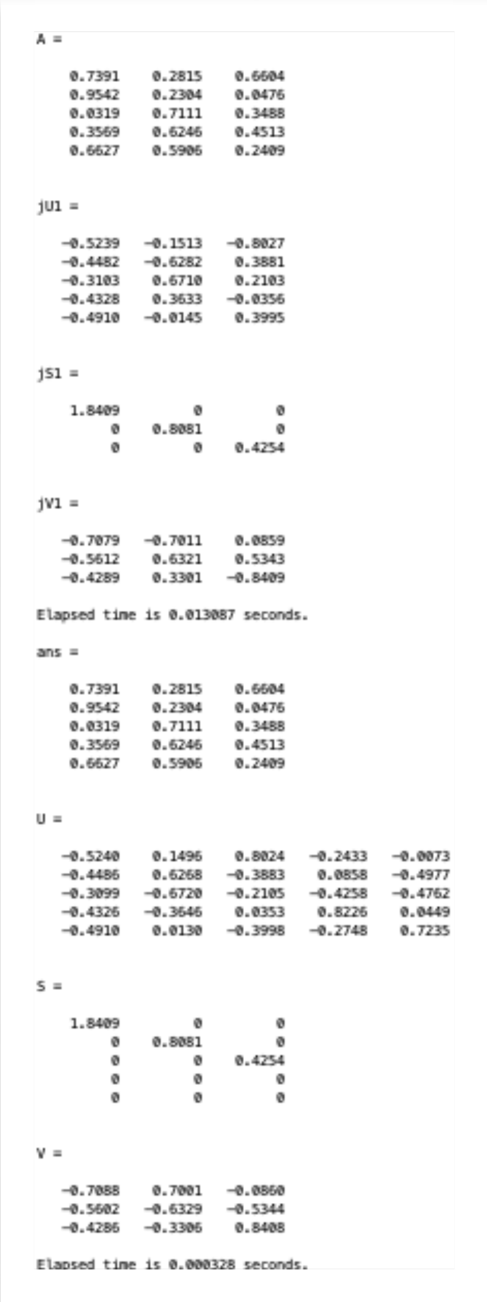
مقایسه روش با svd در ماتریس های رندوم با ابعاد مختلف : (خروجی به فرمت Thin SVD میباشد)



```
A =

    0.7505    0.5836    0.7196
    0.5835    0.5118    0.9962
    0.5518    0.0826    0.3545


jU1 =

    0.6494   -0.1894   -0.7365
    0.6843    0.5682    0.4572
    0.3319   -0.8008    0.4985


jS1 =

    1.8223         0         0
         0    0.3131         0
         0         0    0.1857


jV1 =

    0.5870   -0.8065   -0.0588
    0.4152    0.3645   -0.8329
    0.6950    0.4655    0.5503

Elapsed time is 0.016255 seconds.

ans =

    0.7505    0.5836    0.7196
    0.5835    0.5118    0.9962
    0.5518    0.0826    0.3545


U =

   -0.6491    0.1896   -0.7367
   -0.6845   -0.5680    0.4570
   -0.3318    0.8009    0.4985


S =

    1.8223         0         0
         0    0.3131         0
         0         0    0.1857


V =

   -0.5870    0.8074   -0.0601
   -0.4152   -0.3639   -0.8338
   -0.6950   -0.4645    0.5488

Elapsed time is 0.000397 seconds.
```



```
A =

    0.8944    0.9274    0.6183    0.1248    0.8332
    0.1375    0.9175    0.3433    0.7306    0.3983
    0.3900    0.7136    0.9360    0.6465    0.7498


jU1 =

    0.6294   -0.7107    0.3143
    0.4689    0.6698    0.5757
    0.6197    0.2150   -0.7548


jS1 =

    2.4921         0         0
         0    0.7358         0
         0         0    0.4344


jV1 =

    0.3488   -0.6248    0.1518
    0.5843    0.1480    0.6471
    0.4535   -0.0113   -0.7241
    0.3297    0.7334   -0.0648
    0.4718   -0.2231   -0.1722

Elapsed time is 0.011936 seconds.

ans =

    0.8944    0.9274    0.6183    0.1248    0.8332
    0.1375    0.9175    0.3433    0.7306    0.3983
    0.3900    0.7136    0.9360    0.6465    0.7498


U =

   -0.6294   -0.7107    0.3143
   -0.4689    0.6698    0.5757
   -0.6197    0.2150   -0.7548


S =

    2.4921         0         0         0         0
         0    0.7358         0         0         0
         0         0    0.4344         0         0


V =

   -0.3488   -0.6248    0.1518    0.5542   -0.3973
   -0.5843    0.1480    0.6471   -0.4546   -0.1066
   -0.4535   -0.0113   -0.7241   -0.3648   -0.3698
   -0.3297    0.7334   -0.0648    0.5868   -0.0701
   -0.4718   -0.2231   -0.1722    0.0939    0.8301

Elapsed time is 0.000398 seconds.
```



```
A =

    0.7391    0.2815    0.6604
    0.9542    0.2304    0.0476
    0.0319    0.7111    0.3488
    0.3569    0.6246    0.4513
    0.6627    0.5906    0.2409


jU1 =

   -0.5239   -0.1513   -0.8027
   -0.4482   -0.6282    0.3881
   -0.3103    0.6710    0.2103
   -0.4328    0.3633   -0.0356
   -0.4910   -0.0145    0.3995


jS1 =

    1.8409         0         0
         0    0.8081         0
         0         0    0.4254


jV1 =

   -0.7079   -0.7011    0.0859
   -0.5612    0.6321    0.5343
   -0.4289    0.3301   -0.8409

Elapsed time is 0.013087 seconds.

ans =

    0.7391    0.2815    0.6604
    0.9542    0.2304    0.0476
    0.0319    0.7111    0.3488
    0.3569    0.6246    0.4513
    0.6627    0.5906    0.2409


U =

   -0.5240    0.1496    0.8024   -0.2433   -0.0073
   -0.4486    0.6268   -0.3883    0.0858   -0.4977
   -0.3099   -0.6720   -0.2105   -0.4258   -0.4762
   -0.4326   -0.3646    0.0353    0.8226    0.0449
   -0.4910    0.0130   -0.3998   -0.2748    0.7235


S =

    1.8409         0         0
         0    0.8081         0
         0         0    0.4254
         0         0         0
         0         0         0


V =

   -0.7088    0.7001   -0.0860
   -0.5602   -0.6329   -0.5344
   -0.4286   -0.3306    0.8408

Elapsed time is 0.000328 seconds.
```