

Introduction

QR decomposition also known as QR factorization or QU factorization is used to solve linear least squares problem and the basics of eigen value algorithm. An square matrix can be decomposed as

$$A=QR$$

Where A :- Input matrix

Q :- Orthogonal matrix

R :- Upper triangular matrix

If A is a complex square matrix then there Q is an unitary matrix($Q^T * Q = Q * Q^T = I$)

Where I is identity matrix.

Q is responsible for the triangular form of R.

Method:

In project I have used Householder reflection algorithm . It takes a vector and reflects it on to the hyperplane.

Int this algorithm the Q Is calculated as $Q=I - 2VV^T$

Where V is calculated as $V=U/||U||$

$||U||$ is norm of U matrix

U is calculated as $U=X[0] - \alpha e_1$

Where e is the vector like this $(1\ 0\ 0\ 0\ \dots)^T$

$X[0]$ is the zeroth column in the matrix

First we multiply with the householder matrix with Q_1 then we obtain when we choose the first column matrix for X . The result in a matrix Q_1A with zeros in the left column.

$$Q_1 A = \begin{bmatrix} \alpha_1 & \star & \dots & \star \\ 0 & & & \\ \vdots & & A' & \\ 0 & & & \end{bmatrix}$$

This can be repeated for A' (Q_1A gives the output by deleting the first column),it will give Q_2 ...etc this will continue until the minor matrix will goes to $1*1$.

The R matrix can be calculated as

$$R = Q_t \cdots Q_2 Q_1 A$$

This method gives numerical stability than other decomposition algorithms.

R-matrix parallelization:

Implementation :

In R-Matrix parallelization at first Q matrix is calculated whenever Q matrix calculation is completed the q-flag in the code will get updated and the other cores will start doing R matrix parallelized way.

In householder algorithm the R matrix depends on Q matrix so at first we need to calculate Q matrix to find R matrix. So I first calculated Q matrix and then I am storing the Q matrix values in the shared memory and I have used delay in the code because the shared memory is taking some time to copy the values from one core to other core .

Input matrix I took

```
double input[16] = {
    5.124575, 6.716550, 4.749593, 0.745557, 0.835284, 1.061532, 2.406306, 3.394185, 7.122869, 4.096635, 7.461806, 5.104174, 0.995246, 3.524249, 1.343407, 7.898948,
    4.709401, 5.185006, 2.168096, 1.438314, 3.459950, 4.195184, 6.530458, 4.538856, 1.076255, 2.883427, 6.272343, 5.829543, 2.964446, 0.192133, 7.946213, 1.210326,
    8.103001, 6.291060, 5.034810, 2.737167, 9.811339, 2.894305, 0.413352, 2.290696, 3.690015, 6.639588, 5.670019, 4.373600, 6.214067, 3.592056, 1.200568, 0.145697,
    6.742811, 6.390177, 2.079934, 8.330166, 6.574254, 0.563682, 4.178735, 4.793037, 9.528414, 8.307987, 2.311288, 6.081988, 3.150382, 9.572444, 0.373962, 3.956355,
    5.678259, 8.243491, 5.616988, 8.213211, 1.015888, 1.158368, 1.437121, 6.370580, 2.500651, 7.146724, 9.152067, 4.568377, 4.722142, 2.252590, 6.428624, 5.997973,
    9.094666, 3.641620, 5.956050, 1.629081, 6.459073, 9.468715, 4.752188, 7.708865, 2.220713, 2.484793, 6.529717, 2.083286, 0.599322, 8.827311, 5.588702, 3.405092,
    5.301290, 2.267589, 0.140771, 1.161778, 0.592439, 9.798718, 4.314055, 3.972274, 9.824009, 9.260374, 6.768367, 5.079985, 5.533086, 1.848284, 3.462888, 0.671444,
    9.660500, 7.457226, 2.946092, 4.250254, 2.801475, 0.722540, 7.356618, 6.927009, 8.416201, 6.432596, 7.406141, 4.093572, 9.802859, 9.046292, 8.103308, 0.099300,
    9.001511, 1.054377, 0.355802, 4.236591, 6.461459, 4.018142, 2.400816, 3.651846, 4.161861, 8.746218, 9.976361, 4.284640, 7.141063, 7.410219, 6.350062, 7.652938,
    4.751442, 8.985795, 9.131959, 0.385422, 6.837505, 9.114348, 0.238815, 8.198015, 8.454410, 2.955473, 7.821795, 0.187684, 6.068312, 3.552302, 2.452192, 3.747190,
    7.072122, 1.099953, 0.179495, 1.086232, 9.648564, 4.431138, 2.433835, 4.077998, 4.805562, 1.784756, 2.986027, 2.828004, 8.104572, 1.811495, 0.483971, 3.724873,
    4.902166, 7.299483, 6.822000, 8.115096, 3.138024, 8.902470, 5.816616, 1.066501, 0.368879, 1.762015, 8.535061, 6.834885, 1.670396, 9.097155, 7.376048, 4.097865,
    0.370360, 3.752868, 6.622229, 5.945147, 8.541925, 8.859654, 3.349494, 2.638748, 1.283802, 9.154308, 8.949241, 9.176734, 9.049492, 8.284714, 4.103983, 9.309067,
    8.626863, 3.556053, 0.016668, 6.758221, 5.183024, 8.069129, 0.308171, 5.894849, 4.018287, 6.069993, 7.510052, 0.099244, 8.835959, 4.522346, 4.287552, 8.246637,
    2.015325, 9.879946, 7.386629, 2.522771, 1.881512, 0.924535, 3.546607, 8.241838, 1.206717, 9.042177, 5.350417, 4.953636, 5.543465, 4.732425, 4.703988, 3.690862,
    0.150104, 1.012173, 6.372757, 4.470197, 8.672633, 9.264992, 5.326238, 1.308067, 6.909482, 6.954018, 5.243287, 0.047543, 5.374593, 4.245235, 9.933439, 8.686286,
};
```

Results:

Arm processors results

4 X 4 matrix output :

Q	0.578	0.560	-0.301	0.512
	0.094	0.083	0.929	0.349
	0.803	-0.504	0.079	-0.308
	0.112	0.652	0.201	-0.722
R	8.870	7.665	9.113	5.735
	0.000	4.085	-0.023	3.280
	-0.000	-0.000	1.669	4.924
	0.000	-0.000	0.000	-5.712

8 X 8 matrix output :

Q	0.288	0.396	0.034	-0.505	-0.399	-0.543	-0.065	0.213
	0.400	-0.466	0.611	0.026	-0.333	-0.003	-0.075	-0.365
	0.264	0.172	-0.166	-0.231	0.034	0.189	0.781	-0.416
	0.060	0.369	0.558	0.415	0.003	0.091	0.372	0.479
	0.459	-0.228	0.110	-0.320	0.728	-0.032	-0.041	0.299
	0.208	0.633	0.159	0.013	0.202	0.320	-0.474	-0.403
	0.379	0.046	-0.280	0.629	0.153	-0.562	0.010	-0.203
	0.535	-0.075	-0.414	0.144	-0.366	0.491	-0.125	0.354
R	17.811	16.249	10.817	11.561	11.705	10.287	5.674	10.369
	0.000	5.157	3.460	2.395	3.327	0.480	5.265	-1.601
	-0.000	-0.000	7.776	2.305	0.637	-1.621	3.183	2.201
	-0.000	0.000	-0.000	7.138	1.564	-0.506	3.171	0.785
	-0.000	0.000	0.000	0.000	7.717	-1.975	-0.098	-2.804
	0.000	0.000	0.000	0.000	-0.000	5.666	-1.120	-1.726
	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	7.171	2.585
	0.000	-0.000	-0.000	-0.000	0.000	-0.000	0.000	-2.457

16 X 16 output :

Q	0.288	0.396	0.034	-0.505	-0.399	-0.543	-0.065	0.213
	0.400	-0.466	0.611	0.026	-0.333	-0.003	-0.075	-0.365
	0.264	0.172	-0.166	-0.231	0.034	0.189	0.781	-0.416
	0.060	0.369	0.558	0.415	0.003	0.091	0.372	0.479
	0.459	-0.228	0.110	-0.320	0.728	-0.032	-0.041	0.299
	0.208	0.633	0.159	0.013	0.202	0.320	-0.474	-0.403
	0.379	0.046	-0.280	0.629	0.153	-0.562	0.010	-0.203
	0.535	-0.075	-0.414	0.144	-0.366	0.491	-0.125	0.354
R	17.811	16.249	10.817	11.561	11.705	10.287	5.674	10.369
	0.000	5.157	3.460	2.395	3.327	0.480	5.265	-1.601
	-0.000	-0.000	7.776	2.305	0.637	-1.621	3.183	2.201
	-0.000	0.000	-0.000	7.138	1.564	-0.506	3.171	0.785
	-0.000	0.000	0.000	0.000	7.717	-1.975	-0.098	-2.804
	0.000	0.000	0.000	0.000	-0.000	5.666	-1.120	-1.726
	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	7.171	2.585
	0.000	-0.000	-0.000	-0.000	0.000	-0.000	0.000	-2.457

R-Matrix results:

1 core results:

4 X 4 output :

Total cycles=2319349253

```
Q Matrix
0.577714    0.094165    0.802989    0.112198
0.560173    0.083169    -0.503901    0.652206
-0.300873    0.928780    0.079416    0.201336
0.511790    0.348688    -0.308197    -0.722151
R matrix
8.870442    7.665168    9.112971    5.735174
0.000000    4.084952    -0.023112    3.279678
-0.000000    -0.000000    1.668970    4.923829
0.000000    -0.000000    0.000000    -5.712243
completeddddd
```

8 X 8 output:

Total cycles = 326468851

```
Q Matrix
0.287714    0.399906    0.264404    0.060425    0.459475    0.207621    0.378568    0.534974
0.395902    -0.465658    0.172353    0.368756    -0.227816    0.633346    0.046337    -0.074587
0.034396    0.610537    -0.165686    0.558488    0.109705    0.158516    -0.279886    -0.413766
-0.505463    0.026423    -0.231055    0.414771    -0.319708    0.012762    0.628660    0.144211
-0.399221    -0.332555    0.033514    0.003392    0.728457    0.201636    0.153425    -0.366127
-0.543035    -0.003416    0.189351    0.091020    -0.031511    0.320081    -0.562180    0.491399
-0.064808    -0.074929    0.780662    0.372474    -0.040913    -0.474006    0.009654    -0.124771
0.212937    -0.365466    -0.416097    0.478948    0.299027    -0.403344    -0.203250    0.353767
R matrix
17.811360    16.248665    10.816890    11.561239    11.704696    10.287237    5.674191    10.368926
0.000000    5.156791    3.460376    2.394935    3.326864    0.480324    5.265464    -1.600962
-0.000000    -0.000000    7.775812    2.305135    0.636525    -1.621480    3.182859    2.200535
-0.000000    0.000000    -0.000000    7.138436    1.564049    -0.506463    3.171285    0.785185
-0.000000    0.000000    0.000000    0.000000    7.716916    -1.974703    -0.098065    -2.804130
0.000000    0.000000    0.000000    0.000000    -0.000000    5.666267    -1.120062    -1.725626
-0.000000    -0.000000    -0.000000    -0.000000    -0.000000    -0.000000    7.171003    2.584629
0.000000    -0.000000    -0.000000    -0.000000    0.000000    -0.000000    0.000000    -2.456694
completeddddd
```

16 X 16 output:

Total cycles = 531967459

```
Q Matrix
0.217758 0.200116 0.347757 0.286521 0.156300 0.386458 0.225267 0.410502 0.000064 0.201902 0.300515 0.208307 0.015738 0.366580 0.085637 0.006378
0.184021 0.100587 -0.013922 0.072287 0.369610 -0.247932 -0.134465 -0.014278 0.073091 0.362048 -0.312983 0.236720 0.240061 -0.228118 0.574725 0.061982
-0.032286 -0.172593 0.100097 -0.258518 -0.114184 0.473825 -0.099095 -0.209931 -0.063190 0.216566 0.052790 0.131822 0.363208 -0.171362 -0.113067 0.596450
-0.261506 -0.128975 -0.133621 0.360859 0.306811 -0.162199 -0.052300 -0.049629 0.306129 -0.393615 -0.048329 0.300135 0.300674 0.312801 -0.194674 0.271605
-0.143898 0.188042 0.305169 0.179550 -0.273381 -0.299802 -0.142543 -0.160448 0.451323 0.200450 0.431115 -0.349068 0.199506 -0.044236 0.092739 0.076069
-0.100802 0.248309 -0.321544 -0.256427 -0.175689 -0.094866 0.660007 -0.235920 0.192456 0.271444 -0.066971 0.165168 0.107972 0.270886 -0.001561 -0.019398
0.022413 0.448003 -0.283741 0.074203 -0.244265 0.067568 0.120903 0.409508 0.108708 -0.372972 0.051827 0.069133 0.012072 -0.457979 0.126069 0.286455
-0.162199 -0.115923 -0.462168 -0.011057 0.293779 0.415488 -0.055746 0.095644 0.246334 0.079128 0.048625 -0.542446 -0.039522 0.141316 0.288472 0.012239
0.346154 -0.404952 -0.066154 0.327862 0.008389 -0.255052 0.363234 0.162609 0.027838 0.239254 -0.090229 -0.268506 -0.154200 -0.148804 -0.237332 0.369616
0.104526 -0.162329 0.460488 -0.198543 0.024017 0.044365 0.399340 -0.018814 0.073710 -0.473410 -0.253942 -0.300669 0.274413 0.022191 0.297527 -0.011267
0.438242 0.057771 0.010550 -0.475470 0.238494 -0.019134 -0.163377 0.180011 0.551262 -0.011919 0.007223 0.037834 0.023132 -0.045228 -0.375235 -0.114025
0.221555 -0.065352 -0.200315 0.285630 0.118108 0.183841 0.176322 -0.250562 -0.089930 -0.053414 0.271235 -0.002678 0.469457 -0.367255 -0.159500 -0.467002
-0.190497 -0.238696 -0.123833 -0.342286 0.040222 -0.353856 0.015190 0.515248 -0.106366 0.069615 0.336593 -0.023828 0.400925 0.073500 0.034329 -0.033487
-0.026619 -0.421116 -0.061611 0.139601 -0.570132 0.164424 -0.121366 0.268994 0.329253 0.104147 -0.266096 0.228249 0.126642 0.039200 0.126521 -0.289403
-0.576229 0.122662 0.270491 0.052508 0.226319 0.093626 0.164873 0.235240 0.082489 0.248363 -0.319693 -0.074802 0.035779 -0.338294 -0.340089 -0.157848
0.209274 0.380411 -0.091457 0.131713 -0.175632 -0.009276 -0.231466 0.095512 -0.237619 0.073217 -0.422682 -0.350373 0.410335 0.316362 -0.245003 0.020410
R matrix
23.533358 18.667270 12.337013 12.622347 17.293321 16.876626 11.719547 17.581195 17.568414 17.654125 21.673595 12.865269 18.013469 18.073517 13.986222 11.834372
completedddd
```

2 cores results:

4 X 4 output :

```
row:0 0, 0 1
shm flag0=1,shm.flag1=1 ,shm.flag3=0
total_cycles 0=1299338157 ,total_cycles =1289349253
Q Matrix
0.577714 0.094165 0.802989 0.112198
0.560173 0.083169 -0.503901 0.652206
-0.300873 0.092878 0.079416 0.201336
0.511790 0.348688 -0.308197 -0.722151
R matrix
8.870442 7.665168 9.112971 5.735174
0.000000 4.084952 -0.023112 3.279678
-0.000000 -0.000000 1.668970 4.923829
0.000000 -0.000000 0.000000 -5.712243
completeddddd
```

8 X 8 output :

```
row:0 0, 0 1
shm flag0=1,shm.flag1=1 ,shm.flag3=0
total_cycles 0=2186467949 ,total_cycles =2178711437
Q Matrix
0.287714 0.399906 0.264404 0.060425 0.459475 0.207621 0.378568 0.534974
0.395902 -0.465658 0.172353 0.368756 -0.227816 0.633346 0.046337 -0.074587
0.034396 0.610537 -0.165686 0.558488 0.109705 0.158516 -0.279886 -0.413766
-0.505463 0.026423 -0.231055 0.414771 -0.319708 0.012762 0.628660 0.144211
-0.399221 -0.332555 0.033514 0.003392 0.728457 0.201636 0.153425 -0.366127
-0.543035 -0.003416 0.189351 0.091020 -0.031511 0.320081 -0.562180 0.491399
-0.064808 -0.074929 0.780662 0.372474 -0.040913 -0.474006 0.009654 -0.124771
0.212937 -0.365466 -0.416097 0.478948 0.299027 -0.403344 -0.203250 0.353767
R matrix
17.811360 16.248665 10.816890 11.561239 11.704696 10.287237 5.674191 10.368926
0.000000 5.156791 3.460376 2.394935 3.326864 0.480324 5.265464 -1.600962
-0.000000 -0.000000 7.775812 2.305135 0.636525 -1.621480 3.182859 2.200535
0.000000 0.000000 -0.000000 7.138436 1.564049 -0.506463 3.171285 0.785185
-0.000000 0.000000 0.000000 0.000000 7.716916 -1.974703 -0.098065 -2.804130
0.000000 0.000000 0.000000 0.000000 -0.000000 5.666267 -1.120062 -1.725626
-0.000000 -0.000000 -0.000000 -0.000000 -0.000000 -0.000000 7.171003 2.584629
0.000000 -0.000000 -0.000000 -0.000000 0.000000 -0.000000 0.000000 -2.456694
completeddddd
```

16 X 16 output :

```
row=0 0 0 1
shm_flag=1,shm_flag1=1,shm_flag3=0
total_cycles 0=4294967379,total_cycles=4294967379
Q Matrix
0.217758 0.200116 0.347757 0.286521 0.156300 0.386458 0.225267 0.410502 0.000064 0.201902 0.300515 0.208307 0.015738 0.366580 0.085637 0.006378
0.184021 0.100597 -0.013922 0.072837 0.369610 -0.247932 -0.134465 -0.014278 0.073091 0.362048 -0.312983 0.236720 0.240061 -0.228118 0.574725 0.061982
-0.032258 -0.172593 0.100097 -0.258518 -0.114184 0.473825 -0.089099 -0.208931 -0.063190 0.216566 0.052790 0.131822 0.363208 -0.171362 -0.113067 0.596450
-0.261506 -0.129875 -0.133621 0.360859 0.306811 -0.162199 -0.052800 -0.049629 0.306129 -0.393615 -0.048329 0.300135 0.300674 0.312801 -0.194674 0.271605
-0.143898 0.188042 0.305169 0.179550 -0.273381 -0.299802 -0.142543 -0.160448 0.451323 0.200450 0.431115 -0.349068 0.199906 -0.044236 0.092739 0.076069
-0.100802 0.248309 -0.321544 -0.256427 -0.175689 -0.094866 0.660007 -0.235920 0.192456 0.271444 -0.066971 0.165168 0.107972 0.270886 -0.001561 -0.019398
0.022413 0.448093 -0.283741 0.074203 -0.244265 0.067568 0.120903 0.409508 0.108708 -0.372972 0.051827 0.069133 0.012072 -0.457979 0.126069 0.286455
-0.182199 -0.115923 -0.462168 -0.011057 0.293779 0.415488 -0.055746 0.095644 0.246334 0.079128 0.048625 -0.542446 -0.039522 0.141316 0.288472 0.012239
0.346154 -0.404952 -0.086154 0.327862 0.008389 -0.255052 0.363234 0.163609 0.027838 0.239254 -0.090229 -0.268506 -0.154300 -0.149804 -0.237332 0.369616
0.104526 -0.162329 0.460488 -0.198543 0.024017 0.044365 0.399340 -0.018814 0.073710 -0.473410 -0.253942 -0.300669 0.274413 0.022191 0.297527 -0.011267
0.438242 0.057771 0.010550 -0.475470 0.238494 -0.019134 -0.163377 0.180011 0.551262 -0.011919 0.007223 0.037834 0.023132 -0.045228 -0.375235 -0.114825
0.221555 -0.065352 -0.200315 0.208430 0.110108 0.103841 0.176322 -0.250562 -0.089930 -0.053414 0.271235 -0.002678 0.469457 -0.367255 -0.159500 -0.467002
-0.190497 -0.238696 -0.123833 -0.343286 0.040222 -0.353856 0.015190 0.515248 -0.304366 0.069615 0.336593 -0.023828 0.400825 0.073500 0.034329 -0.033487
-0.026619 -0.421116 -0.061611 0.139601 -0.570133 0.164424 -0.121366 0.268994 0.329253 0.104147 -0.266096 0.228249 0.126642 0.039200 0.126521 -0.289403
-0.576229 0.122662 0.270491 0.052508 0.226319 0.093626 0.164873 0.239240 0.082489 0.248363 -0.319693 -0.074802 0.035779 -0.338294 -0.340089 -0.157848
0.209274 0.380411 -0.091457 0.131713 -0.175632 -0.009276 -0.231466 0.095512 -0.237619 0.073217 -0.422482 -0.350373 0.410335 0.316362 -0.245003 0.020410
R Matrix
23.533358 18.667270 12.337613 12.622347 17.293321 16.876626 11.719547 17.581195 17.568414 17.654125 21.673595 12.865269 18.013469 18.073517 13.966222 11.834372
0.000000 14.409226 12.827850 6.903432 2.652707 3.075349 4.043177 5.703948 3.450561 10.256326 11.322193 8.626997 5.589017 6.941668 7.737697 7.866439
0.000000 0.000000 9.219503 9.988264 10.125890 13.010211 2.606845 0.909640 1.150138 1.801833 5.302148 0.356005 2.264915 5.404558 5.762875 7.191222
-0.000000 -0.000000 0.000000 12.384407 5.428370 4.873513 2.181349 -0.513640 1.521957 8.389030 6.149520 5.224770 6.006798 8.400323 6.587891 9.233891
-0.000000 -0.000000 0.000000 0.000000 11.620378 1.627563 -0.314418 1.978190 4.149088 6.037508 2.368590 2.522606 8.735393 1.238416 -0.653910 3.935253
0.000000 0.000000 0.000000 0.000000 0.000000 12.407613 2.243859 2.623311 3.691292 4.277892 7.025525 2.107866 4.294928 -0.268594 4.522370 3.493647
-0.000000 -0.000000 -0.000000 0.000000 -0.000000 0.000000 5.248935 0.472587 2.353381 2.812587 1.353896 4.872543 0.740814 3.868053 7.633962 -0.949284
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 7.582893 1.355163 3.704564 1.907089 -2.679734 3.116815 1.658664 2.733680 3.567886
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 12.308383 4.662618 0.144035 -1.466594 1.563744 0.830611 -1.309500 2.273941
0.000000 0.000000 -0.000000 -0.000000 0.000000 -0.000000 0.000000 0.000000 8.748723 2.850105 4.004529 3.385391 0.508259 0.474221 0.841942
0.000000 -0.000000 -0.000000 0.000000 -0.000000 -0.000000 0.000000 0.000000 0.000000 7.886933 1.762301 2.248908 0.867585 3.951856 4.789101
0.000000 0.000000 -0.000000 -0.000000 -0.000000 -0.000000 -0.000000 0.000000 0.000000 0.000000 -0.000000 6.263494 -2.294266 1.836599 -5.632187 0.978521
-0.000000 0.000000 0.000000 0.000000 -0.000000 -0.000000 -0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 7.575371 -0.746620 0.070973 -0.791225
0.000000 0.000000 -0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 8.300289 -2.086359 -0.785619
-0.000000 0.000000 0.000000 0.000000 0.000000 -0.000000 -0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 1.009370 -7.368947
0.000000 0.000000 0.000000 0.000000 0.000000 -0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 -0.000000 0.000000 0.000000 2.524540
completedddd
```

Q – Matrix parallelization :-

Implementation :

In Q matrix parallelization

we know that R can be calculated as

$$R = Q_t \cdots Q_2 Q_1 A$$

In householder algorithm we will calculate $Q = (Q_1 * Q_2 * \dots * Q_n)$. We know the commutative formula in matrices $(A * B * C = A * (B * C))$ OR $(A * B) * C$ like this I taught to calculate Q also $Q_c = (Q_1 * Q_2 * \dots * Q_a)$ in one core and $Q_d = (Q_a * Q_{a+1} * \dots * Q_n)$ in other core and multiplying $Q_c * Q_d$ finally, but while doing like this my flags are not getting updated properly I don't know the reason why. But when trying to calculate Q_c or Q_d in single core the values and flags are coming properly. Externally I tried to do multiplication of Q_c and Q_d I got exact Q matrix what I did in serial execution.

I took 4 X 4 matrix to do this

$Q_c = Q_1 * Q_2$

```
2
0.578 0.560 0.473 -0.359
0.094 0.083 0.456 0.881
0.803 -0.504 -0.297 0.115
0.112 0.652 -0.693 0.285
```

$$Q_d = Q_3 * Q_4$$

1.000	0.000	0.000	0.000
0.000	1.000	0.000	0.000
0.000	0.000	0.984	-0.177
0.000	0.000	0.177	0.984

$$Q = Q_c * Q_d$$

0.578	0.560	-0.301	0.512
0.094	0.083	0.929	0.349
0.803	-0.504	0.079	-0.308
0.112	0.652	0.201	-0.722

Conclusion:

In householder matrix the R matrix is nothing but multiplication of Q and input matrix(A)

While doing R matrix parallelization the computation time is decreased by 1000000000 nearly ,in R matrix parallelization the Q matrix is calculated in sequential manner after calculation of Q matrix the q_flag will get updated to 1 . whenever the q_flag updates to 1 the final multiplication of Q and A is done parallely .

While performing Q matrix parallelization the flags are not updated but I when I calculated the cycles of Qc or Qd the cycles decreased by half.

When I tried to increase number of cores from 2 to 4 the flags are not updating at all. I don't know why this is happening like that.

References:

https://en.wikipedia.org/wiki/QR_decomposition#Using_Householder_reflections