NA15 QR rastav

Ivan Slapničar

3. prosinca 2018.

1 QR rastav

QR rastav matrice A tipa $m \times n$, $m \geq n$, glasi

$$A = QR,$$

pri čemu je Q *ortonormirana matrica* dimenzije $m \times m$, odnosno

$$Q^T Q = QQ^T = I,$$

a R je $m \times n$ gornje trokutasta matrica.
Ortonormiranu matricu kraće zovemo i *ortogonalna matrica*.
Na primjer,

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \\ a_{51} & a_{52} & a_{53} \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} & q_{15} \\ q_{21} & q_{22} & q_{23} & q_{24} & q_{25} \\ q_{31} & q_{32} & q_{33} & q_{34} & q_{35} \\ q_{41} & q_{42} & q_{43} & q_{44} & q_{45} \\ q_{51} & q_{52} & q_{53} & q_{54} & q_{55} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{1}$$

S (1) je definiram i *ekonomični QR rastav*

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \\ a_{51} & a_{52} & a_{53} \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \\ q_{41} & q_{42} & q_{43} \\ q_{51} & q_{52} & q_{53} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix}. \tag{2}$$

Izjednačavanje stupaca počevši od prvog daje:

$$
\begin{aligned}
t &= a_{:1} \\
r_{11} &= \|t\|_2 \\
q_{:1} &= t\frac{1}{r_{11}} \\
r_{12} &= q_{:1}^T a_{:2} \\
t &= a_{:2} - q_{:1} r_{12} \\
r_{22} &= \|t\|_2 \\
q_{:2} &= t\frac{1}{r_{22}} \\
r_{13} &= q_{:1}^T a_{:3} \\
r_{23} &= q_{:2}^T a_{:3} \\
t &= a_{:3} - q_{:1} r_{13} - q_{:2} r_{23} \\
r_{33} &= \|t\|_2 \\
q_{:3} &= t\frac{1}{r_{33}}.
\end{aligned}
$$

Indukcijom slijedi *Gram-Schmidtov postupak ortogonalizacije*.

```
In [1]: using LinearAlgebra
        function myGramSchmidtQR(A::Array)
            m,n=size(A)
            R=zeros(n,n)
            Q=Array{Float64}(undef,m,n)
            R[1,1]=norm(A[:,1])
            Q[:,1]=A[:,1]/R[1,1]
            for k=2:n
                for i=1:k-1
                    R[i,k]=Q[:,i]⋅A[:,k]
                end
                t=A[:,k]-sum([R[i,k]*Q[:,i] for i=1:k-1])
                R[k,k]=norm(t)
                Q[:,k]=t/R[k,k]
            end
            Q,R
        end
```

```
Out[1]: myGramSchmidtQR (generic function with 1 method)
```

```
In [2]: import Random
        Random.seed!(123)
        A=rand(8,5)
```

```
Out[2]: 8×5 Array{Float64,2}:
         0.768448   0.26864   0.275819  0.20923   0.356221
```

```
 0.940515   0.108871  0.446568  0.918165  0.900925
 0.673959   0.163666  0.582318  0.614255  0.529253
 0.395453   0.473017  0.255981  0.802665  0.031831
 0.313244   0.865412  0.70586   0.555668  0.900681
 0.662555   0.617492  0.291978  0.940782  0.940299
 0.586022   0.285698  0.281066  0.48      0.621379
 0.0521332  0.463847  0.792931  0.790201  0.348173
```

```
In [3]: Q,R=myGramSchmidtQR(A)
```

```
Out[3]: ([0.445979 -0.112473 ... -0.581217 -0.414126; 0.545841 -0.354798 ...
0.296518 0.303573; ... ; 0.340106 -0.00596216 ... -0.076101 0.141458;
0.0302562 0.432357 ... 0.25137 -0.128359], [1.72306 0.857781 ...
1.66889 1.61212; 0.0 1.01281 ... 0.760568 0.603988; ... ; 0.0 0.0 ...
0.686493 -0.00271451; 0.0 0.0 ... 0.0 0.652889])
```

```
In [4]: Q
```

```
Out[4]: 8×5 Array{Float64,2}:
 0.445979   -0.112473    -0.144567   -0.581217   -0.414126
 0.545841   -0.354798     0.210356    0.296518    0.303573
 0.391141   -0.169675     0.45213    -0.0982655  -0.123261
 0.229507    0.272659    -0.24854     0.435716   -0.699857
 0.181796    0.700501     0.0463297  -0.432191    0.268038
 0.384523    0.284018    -0.440047    0.344953    0.350741
 0.340106   -0.00596216  -0.0882082  -0.076101    0.141458
 0.0302562   0.432357     0.681975    0.25137    -0.128359
```

```
In [5]: Q'*Q
```

```
Out[5]: 5×5 Array{Float64,2}:
  1.0           7.28584e-17  -3.81639e-17   4.96131e-16   2.77556e-16
  7.28584e-17   1.0          -5.55112e-17  -2.35922e-16  -5.55112e-17
 -3.81639e-17  -5.55112e-17   1.0           1.11022e-16   2.39392e-16
  4.96131e-16  -2.35922e-16   1.11022e-16   1.0          -1.02696e-15
  2.77556e-16  -5.55112e-17   2.39392e-16  -1.02696e-15   1.0
```

```
In [6]: R
```

```
Out[6]: 5×5 Array{Float64,2}:
 1.72306  0.857781  1.01346   1.66889    1.61212
 0.0      1.01281   0.700064  0.760568   0.603988
 0.0      0.0       0.67391   0.349435   0.179984
 0.0      0.0       0.0       0.686493  -0.00271451
 0.0      0.0       0.0       0.0        0.652889
```

```
In [7]: # Rezidual
        A-Q*R
```

```
Out[7]: 8×5 Array{Float64,2}:
         0.0  0.0  0.0  -5.55112e-17   0.0
         0.0  0.0  0.0   0.0          -1.11022e-16
         0.0  0.0  0.0   0.0           0.0
         0.0  0.0  0.0   0.0          -5.55112e-17
         0.0  0.0  0.0   0.0           0.0
         0.0  0.0  0.0   0.0           0.0
         0.0  0.0  0.0   0.0           0.0
         0.0  0.0  0.0   0.0           0.0
```

Algoritam `myGramSchmidtQR()` je numerički nestabilan pa je bolje koristiti *modificirani Gram-Schmidtov algoritam* ili *Householderove reflektore* ili *Givensove rotacije* (vidi Matrix Computations, poglavlje 5).

1.1 Householderovi reflektori

QR rastav vektora x jednak je

$$H \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = r,$$

gdje je

$$H = I - \frac{2}{v^T v} v v^T, \qquad v = \begin{bmatrix} x_1 \pm \|x\|_2 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{bmatrix}.$$

Householderov reflektor H je *simetrična* i *ortogonalna* matrica (dokažite!). Ovisno o izboru predznaka u definicije vektora v vrijedi

$$r = \begin{bmatrix} \mp\|x\| \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Zbog numeričke stabilnost se najčešće uzima

$$v_1 = x_1 + \text{sign}(x_1)\|x\|_2.$$

Matrica H se *ne računa eksplicitno* već se produkt Hx računa po formuli

$$Hx = x - \frac{2(v^T x)}{v^T v} v = x - \frac{2(v \cdot x)}{v \cdot v} v$$

za koju je potrebno $O(6m)$ operacija.

```
In [8]: function myHouseholderVector(x::Array)
            # Racuna v
            v=deepcopy(x)
            v[1]=x[1]+sign(x[1])*norm(x)
            v
        end
```

```
Out[8]: myHouseholderVector (generic function with 1 method)
```

```
In [9]: x=rand(8)
        v=myHouseholderVector(x)
        β=(2/(v⋅v))*(v⋅x)
        x-β*v
```

```
Out[9]: 8-element Array{Float64,1}:
         -1.2568320365216425
          0.0
          0.0
          0.0
          0.0
          0.0
          0.0
          0.0
```

```
In [10]: norm(x)
```

```
Out[10]: 1.2568320365216425
```

QR rastav matrice se računa rekurzivnim QR rastavom vektora pomoću Householderovih reflektora:

```
In [11]: function myHouseholderQR(A1::Array)
             # Racuna Q i R
             A=deepcopy(A1)
             m,n=size(A)
             Q=Matrix{Float64}(I,m,m) # eye
             for k=1:n
                 v=myHouseholderVector(A[k:m,k])
                 β=(2/(v⋅v))*v
                 A[k:m,k:n]=A[k:m,k:n]-β*(v'*A[k:m,k:n])
                 Q[k:m,:]=Q[k:m,:]-β*(v'*Q[k:m,:])
             end
             R=triu(A)
             Q',R
         end
```

```
Out[11]: myHouseholderQR (generic function with 1 method)

In [12]: A

Out[12]: 8×5 Array{Float64,2}:
          0.768448   0.26864   0.275819  0.20923   0.356221
          0.940515   0.108871  0.446568  0.918165  0.900925
          0.673959   0.163666  0.582318  0.614255  0.529253
          0.395453   0.473017  0.255981  0.802665  0.031831
          0.313244   0.865412  0.70586   0.555668  0.900681
          0.662555   0.617492  0.291978  0.940782  0.940299
          0.586022   0.285698  0.281066  0.48      0.621379
          0.0521332  0.463847  0.792931  0.790201  0.348173

In [13]: Q,R=myHouseholderQR(A)

Out[13]: ([-0.445979 -0.112473 ... -0.184609 0.445399; -0.545841 -0.354798 ...
-0.284531 0.161726; ... ; -0.340106 -0.00596216 ... 0.911612 0.0807266;
-0.0302562 0.432357 ... 0.0808767 0.47278], [-1.72306 -0.857781 ...
-1.66889 -1.61212; 0.0 1.01281 ... 0.760568 0.603988; ... ;
0.0 0.0 ... 0.0 0.0; 0.0 0.0 ... 0.0 0.0])

In [14]: Q'*A

Out[14]: 8×5 Array{Float64,2}:
          -1.72306      -0.857781     -1.01346      -1.66889      -1.61212
           1.45717e-16   1.01281       0.700064      0.760568      0.603988
           2.08167e-17  -5.55112e-17  -0.67391      -0.349435     -0.179984
          -6.07153e-17  -2.77556e-17   8.32667e-17  -0.686493      0.00271451
           1.83881e-16   2.70617e-16   2.91434e-16   1.38778e-16  -0.652889
           2.94903e-17   5.55112e-17   1.11022e-16   1.11022e-16   1.11022e-16
          -2.42861e-17   6.93889e-18   5.55112e-17   4.16334e-17  -1.11022e-16
           3.1225e-17    0.0           5.55112e-17   0.0           0.0

In [15]: R

Out[15]: 8×5 Array{Float64,2}:
          -1.72306  -0.857781  -1.01346   -1.66889   -1.61212
           0.0       1.01281    0.700064   0.760568   0.603988
           0.0       0.0       -0.67391   -0.349435  -0.179984
           0.0       0.0        0.0       -0.686493   0.00271451
           0.0       0.0        0.0        0.0       -0.652889
           0.0       0.0        0.0        0.0        0.0
           0.0       0.0        0.0        0.0        0.0
           0.0       0.0        0.0        0.0        0.0
```

Program `myHouseholderQR()` je ilustrativan. Profesionalni programi imaju sljedeća svojstva:

- računaju s blok matricama (uobičajena dimenzija bloka je 32 ili 64),
- izračuna se vektor $\hat{v} = v/v_1$. Vrijedi $\hat{v}_1 = 1$, dok se ostali elemenenti vektora $\hat{v}$ spremaju u strogi donji trokut matrice A,
- ako se traži matrica Q, akumulacija se vrši unatrag koristeći spremljene vektore v (tako se smanjuje broj operacija),
- postoji opcija vraćanja ekonomičnog rastava,
- postoji opcija računanja s *pivotiranjem* - u svakom koraku se na prvo mjesto dovede stupac s najvećom normom pa je

$$AP = QR.$$

Vrijedi

$$|R_{kk}| \geq |R_{k+1,k+1}|$$

pa se može utvrditi i numerički rank matrice.

```
In [16]: # ?qr
```

```
In [17]: F=qr(A)
```

```
Out[17]: LinearAlgebra.QRCompactWY{Float64,Array{Float64,2}}
         Q factor:
         8×8 LinearAlgebra.QRCompactWYQ{Float64,Array{Float64,2}}:
          -0.445979   -0.112473     0.144567   ...   0.160558  -0.184609    0.445399
          -0.545841   -0.354798    -0.210356       -0.494706  -0.284531    0.161726
          -0.391141   -0.169675    -0.45213         0.385539   0.0148903  -0.66339
          -0.229507    0.272659     0.24854        -0.295448   0.0184688  -0.209603
          -0.181796    0.700501    -0.0463297      -0.362887  -0.160778   -0.240692
          -0.384523    0.284018     0.440047   ...   0.558011  -0.144818    0.0589447
          -0.340106   -0.00596216   0.0882082      -0.114703   0.911612    0.0807266
          -0.0302562   0.432357    -0.681975        0.193228   0.0808767   0.47278
         R factor:
         5×5 Array{Float64,2}:
          -1.72306  -0.857781  -1.01346   -1.66889   -1.61212
           0.0       1.01281    0.700064   0.760568   0.603988
           0.0       0.0       -0.67391   -0.349435  -0.179984
           0.0       0.0        0.0       -0.686493   0.00271451
           0.0       0.0        0.0        0.0       -0.652889
```

```
In [18]: F.Q'*A
```

```
Out[18]: 8×5 Array{Float64,2}:
          -1.72306      -0.857781     -1.01346      -1.66889      -1.61212
          -1.11022e-16   1.01281       0.700064      0.760568      0.603988
          -1.11022e-16   2.77556e-17  -0.67391      -0.349435     -0.179984
```

```
 -2.77556e-16  -1.66533e-16  -3.33067e-16  -0.686493      0.00271451
 -3.88578e-16   1.11022e-16   0.0          -2.22045e-16  -0.652889
 -3.33067e-16  -1.11022e-16  -1.11022e-16  -2.22045e-16   1.11022e-16
  0.0           0.0           0.0           0.0           1.11022e-16
 -1.94289e-16   5.55112e-17  -1.11022e-16  -1.11022e-16  -5.55112e-17
```

In [19]:
```
F.Q*F.R
```

Out[19]:
```
8×5 Array{Float64,2}:
 0.768448   0.26864   0.275819  0.20923   0.356221
 0.940515   0.108871  0.446568  0.918165  0.900925
 0.673959   0.163666  0.582318  0.614255  0.529253
 0.395453   0.473017  0.255981  0.802665  0.031831
 0.313244   0.865412  0.70586   0.555668  0.900681
 0.662555   0.617492  0.291978  0.940782  0.940299
 0.586022   0.285698  0.281066  0.48      0.621379
 0.0521332  0.463847  0.792931  0.790201  0.348173
```

In [20]:
```
F=qr(A,Val(true))
```

Out[20]:
```
QRPivoted{Float64,Array{Float64,2}}
Q factor:
8×8 LinearAlgebra.QRPackedQ{Float64,Array{Float64,2}}:
 -0.105181  -0.657376   -0.16259    ...  -0.418105   -0.280647   -0.0682982
 -0.461568  -0.291448    0.0230722      -0.0183557  -0.262798    0.531458
 -0.30879   -0.242706   -0.109412        0.57546     0.1056     -0.496631
 -0.403505   0.200334   -0.681506        0.246128    0.0857566   0.252251
 -0.279338   0.0965824   0.643895        0.323668   -0.0771281   0.323665
 -0.472938   0.0225186   0.243739    ...  -0.117048   -0.204861   -0.529297
 -0.241299  -0.252973    0.145506       -0.246011    0.885934    0.0728644
 -0.397239   0.556817   -0.0378098      -0.504117   -0.0295477  -0.111318
R factor:
5×5 Array{Float64,2}:
 -1.98923  -1.44558   -1.61412   -1.10689   -1.2363
  0.0      -0.937667  -0.473979   0.130204   0.0436452
  0.0       0.0        0.76965    0.350337   0.263875
  0.0       0.0        0.0       -0.629825  -0.177484
  0.0       0.0        0.0        0.0       -0.582983
permutation:
5-element Array{Int64,1}:
 4
 1
 5
 2
 3
```

In [21]:
```
# Vektor pivotiranja
F.p
```

```
Out[21]: 5-element Array{Int64,1}:
          4
          1
          5
          2
          3

In [22]: # Matrica pivotiranja
         F.P

Out[22]: 5×5 Array{Float64,2}:
          0.0  1.0  0.0  0.0  0.0
          0.0  0.0  0.0  1.0  0.0
          0.0  0.0  0.0  0.0  1.0
          1.0  0.0  0.0  0.0  0.0
          0.0  0.0  1.0  0.0  0.0

In [23]: # Provjera s matricom
         F.Q*F.R-A*F.P

Out[23]: 8×5 Array{Float64,2}:
          0.0          8.88178e-16  4.44089e-16   2.22045e-16  4.44089e-16
          0.0          4.44089e-16  3.33067e-16  -1.11022e-16  1.11022e-16
          0.0          3.33067e-16  1.11022e-16   8.32667e-17  0.0
          0.0          2.77556e-16  1.11022e-16   2.22045e-16  5.55112e-17
          0.0          3.33067e-16  0.0           2.22045e-16  2.22045e-16
          1.11022e-16  4.44089e-16  2.22045e-16   2.22045e-16  3.33067e-16
          5.55112e-17  2.22045e-16  1.11022e-16   0.0          1.11022e-16
          1.11022e-16  1.11022e-16  5.55112e-17   2.77556e-16  2.22045e-16

In [24]: # Provjera s vektorom
         F.Q*F.R-A[:,F.p]

Out[24]: 8×5 Array{Float64,2}:
          0.0          8.88178e-16  4.44089e-16   2.22045e-16  4.44089e-16
          0.0          4.44089e-16  3.33067e-16  -1.11022e-16  1.11022e-16
          0.0          3.33067e-16  1.11022e-16   8.32667e-17  0.0
          0.0          2.77556e-16  1.11022e-16   2.22045e-16  5.55112e-17
          0.0          3.33067e-16  0.0           2.22045e-16  2.22045e-16
          1.11022e-16  4.44089e-16  2.22045e-16   2.22045e-16  3.33067e-16
          5.55112e-17  2.22045e-16  1.11022e-16   0.0          1.11022e-16
          1.11022e-16  1.11022e-16  5.55112e-17   2.77556e-16  2.22045e-16
```

1.2 Brzina

Broj računskih operacija potrebnih za računanje QR rastava matrice $n \times n$ je $O\left(\frac{4}{3}n^3\right)$ za računanje matrice R i $O\left(\frac{4}{3}n^3\right)$ za računanje matrice Q.

```
In [25]: n=512
         A=rand(n,n);

In [26]: @time qr(A);

  0.063898 seconds (12 allocations: 2.282 MiB)


In [27]: @time qr(A,Val(true));

  0.159835 seconds (12 allocations: 2.142 MiB)


In [28]: @time myHouseholderQR(A);

  4.167273 seconds (12.62 k allocations: 3.354 GiB, 10.89% gc time)
```

1.3 Točnost

Za matrice $\hat{Q}$ i $\hat{R}$ izračunate Householder-ovom metodom vrijedi:

$$\hat{Q}^T\hat{Q} = I + E, \qquad \|E\|_2 \approx \varepsilon,$$
$$\|A - \hat{Q}\hat{R}\|_2 \approx \varepsilon\|A\|_2.$$

Također, postoji egzaktna ortogonalna matrica Q za koju je

$$\|A - Q\hat{R}\|_2 \approx \varepsilon\|A\|_2.$$