NA16 Primjene QR rastava

Ivan Slapničar

3. prosinca 2018.

1 Primjene QR rastava

1.1 Rješavanje sustava linearnih jednadžbi

QR rastav možemo koristiti za rješavanje sustava linearnih jednadžbi. U odnosu na rješenje pomoću Gaussove eliminacije vrijedi:

- broj računskih operacija se udvostruči,
- rješenje je nešto točnije, i
- nema rasta elemenata (pivotiranje nije potrebno).

In [1]:
```
import Random
Random.seed!(123)
using LinearAlgebra
n=10
A=rand(n,n)
b=rand(n)
Q,R=qr(A)
c=Q'*b
# Trokutasti sustav
x=R\c
```

Out[1]:
```
10-element Array{Float64,1}:
 -1.3513832062047446
 -2.803574341341646
 -6.212172382452086
 -0.4444920506128095
  2.120799184124583
  4.959355245144356
  1.7098940042037887
  1.1499707113735076
  1.533300587444642
  2.271279372899755
```

In [2]:
```
A*x-b
```

```
Out[2]: 10-element Array{Float64,1}:
          1.5543122344752192e-15
          1.3322676295501878e-15
          7.771561172376096e-16
         -3.1086244689504383e-15
         -4.440892098500626e-16
          1.3322676295501878e-15
         -8.881784197001252e-16
         -1.4432899320127035e-15
         -2.220446049250313e-16
          5.551115123125783e-16
```

1.2 Rješavanje problema najmanjih kvadrata

Programi za rješavanje problema najmanjih kvadrata uglavnom koriste QR rastav. Vrijedi

$$\|Ax-b\|_2^2 = \|QRx-b\|_2^2 = \|Q(Rx-Q^Tb)\|_2 = \|Rx-Q^Tb\|_2^2.$$

Neka je

$$R = \begin{bmatrix} R_0 \\ 0 \end{bmatrix}, \quad Q^Tb = \begin{bmatrix} c \\ d \end{bmatrix}.$$

Tada je

$$\|Rx-Q^Tb\|_2^2 = \|R_0x-c\|_2^2 + \|d\|_2^2$$

pa je rješenje trokutastog sustava

$$R_0x = c$$

rješenje problema najmanjih kvadrata.

```
In [3]: A=[1 2 3 6 7;1 1 1 1 1]'
```

```
Out[3]: 5×2 Adjoint{Int64,Array{Int64,2}}:
         1  1
         2  1
         3  1
         6  1
         7  1
```

```
In [4]: Q,R=qr(A)
```

```
Out[4]: LinearAlgebra.QRCompactWY{Float64,Array{Float64,2}}
        Q factor:
```

```
5×5 LinearAlgebra.QRCompactWYQ{Float64,Array{Float64,2}}:
 -0.100504  -0.694576  -0.36286   -0.423284  -0.443425
 -0.201008  -0.529614  -0.240632   0.433514   0.65823
 -0.301511  -0.364653   0.878914  -0.052476  -0.0296059
 -0.603023   0.130233  -0.135336   0.582037  -0.512172
 -0.703526   0.295195  -0.140086  -0.539791   0.326973
R factor:
2×2 Array{Float64,2}:
 -9.94987  -1.90957
  0.0      -1.16342
```

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

Out[5]:
```
5×5 Array{Float64,2}:
  1.0           -8.32667e-17  -9.71445e-17  -1.66533e-16  -8.32667e-17
 -8.32667e-17   1.0           -6.93889e-17  -1.66533e-16  -1.52656e-16
 -9.71445e-17  -6.93889e-17   1.0           -5.55112e-17  -4.16334e-17
 -1.66533e-16  -1.66533e-16  -5.55112e-17   1.0           -2.77556e-17
 -8.32667e-17  -1.52656e-16  -4.16334e-17  -2.77556e-17   1.0
```

In [6]:
```
m=8
n=5
A=rand(m,n)
b=rand(m)
Q,R=qr(A)
```

Out[6]:
```
LinearAlgebra.QRCompactWY{Float64,Array{Float64,2}}
Q factor:
8×8 LinearAlgebra.QRCompactWYQ{Float64,Array{Float64,2}}:
 -0.0566351  -0.574505    -0.0230314  …  -0.428439    0.498927   -0.316609
 -0.208694   -0.212997     0.342354       0.408418   -0.169411   -0.646288
 -0.065727   -0.527101    -0.139024      -0.106073    0.0272808   0.257775
 -0.22419    -0.00607116   0.32014       -0.616738   -0.468554    0.212567
 -0.793791   -0.0286567   -0.467893       0.0531959  -0.248767   -0.0696813
 -0.378103    0.262243    -0.0385568  …   0.137668    0.611397    0.287264
 -0.342123    0.234304     0.644938      -0.15383     0.245325   -0.0412849
 -0.0924361  -0.471395     0.352264       0.461055   -0.078795    0.530431
R factor:
5×5 Array{Float64,2}:
 -1.23002  -0.827746  -1.40152   -0.968313  -0.828217
  0.0      -1.13251   -0.809317  -0.595303  -0.266157
  0.0       0.0        0.999553   0.588012   0.814772
  0.0       0.0        0.0       -0.661953   0.267255
  0.0       0.0        0.0        0.0        1.11871
```

In [7]:
```
c=Q[:,1:n]'*b
x=R\c
```

```
Out[7]: 5-element Array{Float64,1}:
         -0.1451430564241811
          0.6290700505109131
          0.6734603175535058
          0.1557491169211927
         -0.3955493171917202
```

```
In [8]: x1=A\b
```

```
Out[8]: 5-element Array{Float64,1}:
         -0.14514305642418174
          0.6290700505109127
          0.6734603175535074
          0.15574911692119206
         -0.3955493171917206
```

1.3 Numerička "ortogonalizacija" polinoma

Numerička ortogonalizacija potencija vektora daje ortogonalne polinome.

```
In [9]: # Standardna baza
        n=100
        x=range(-1,stop=1,length=n)
        # Kvazi Vandermondeova matrica
        V=[x.^0 x.^1 x.^2 x.^3 x.^4 x.^5]
```

```
Out[9]: 100×6 Array{Float64,2}:
         1.0  -1.0       1.0       -1.0       1.0       -1.0
         1.0  -0.979798  0.960004  -0.94061   0.921608  -0.902989
         1.0  -0.959596  0.920824  -0.883619  0.847918  -0.813658
         1.0  -0.939394  0.882461  -0.828978  0.778737  -0.731541
         1.0  -0.919192  0.844914  -0.776638  0.713879  -0.656192
         1.0  -0.89899   0.808183  -0.726548  0.65316   -0.587184
         1.0  -0.878788  0.772268  -0.67866   0.596398  -0.524107
         1.0  -0.858586  0.73717   -0.632923  0.543419  -0.466572
         1.0  -0.838384  0.702887  -0.589289  0.494051  -0.414204
         1.0  -0.818182  0.669421  -0.547708  0.448125  -0.366648
         1.0  -0.79798   0.636772  -0.508131  0.405478  -0.323563
         1.0  -0.777778  0.604938  -0.470508  0.36595   -0.284628
         1.0  -0.757576  0.573921  -0.434789  0.329385  -0.249534
         ⋮                                               ⋮
         1.0   0.777778  0.604938   0.470508  0.36595    0.284628
         1.0   0.79798   0.636772   0.508131  0.405478   0.323563
         1.0   0.818182  0.669421   0.547708  0.448125   0.366648
         1.0   0.838384  0.702887   0.589289  0.494051   0.414204
         1.0   0.858586  0.73717    0.632923  0.543419   0.466572
```

```
1.0   0.878788  0.772268   0.67866   0.596398   0.524107
1.0   0.89899   0.808183   0.726548  0.65316    0.587184
1.0   0.919192  0.844914   0.776638  0.713879   0.656192
1.0   0.939394  0.882461   0.828978  0.778737   0.731541
1.0   0.959596  0.920824   0.883619  0.847918   0.813658
1.0   0.979798  0.960004   0.94061   0.921608   0.902989
1.0   1.0       1.0        1.0       1.0        1.0
```

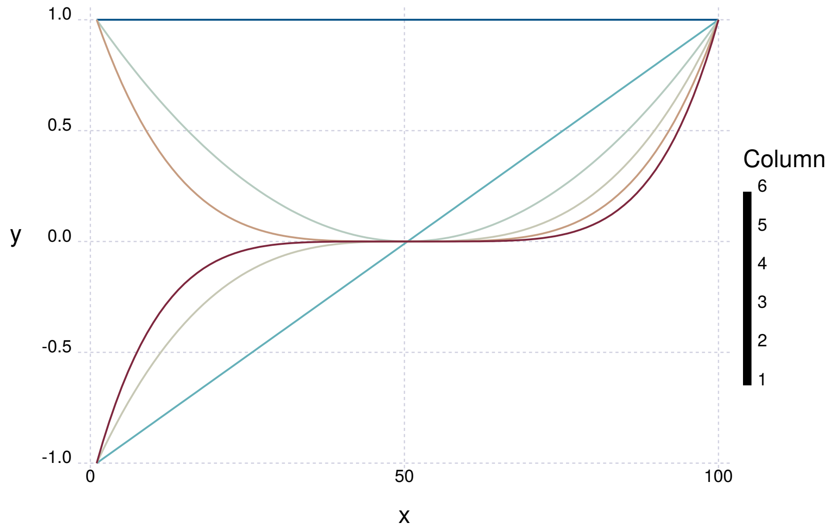
In [10]:
```
# Ortogonalizacija s težinskom funkcijom ω=1 daje normirane
# Legendreove polinome.
Q,R=qr(V)
Q=Q*sign.(diagm(0=>diag(R)))
```

Out[10]:
```
100×6 Array{Float64,2}:
 0.1  -0.171482  0.216998   -0.249164      0.271442   -0.285443
 0.1  -0.168017  0.203846   -0.218963      0.216605   -0.198945
 0.1  -0.164553  0.190963   -0.190302      0.166805   -0.124804
 0.1  -0.161089  0.178349   -0.16315       0.121798   -0.0620008
 0.1  -0.157625  0.166002   -0.137476      0.0813471  -0.00956382
 0.1  -0.15416   0.153925   -0.113247      0.0452209   0.0334314
 0.1  -0.150696  0.142115   -0.0904321     0.0131919   0.0678636
 0.1  -0.147232  0.130574   -0.0689992    -0.014962    0.0945669
 0.1  -0.143767  0.119302   -0.0489166    -0.0394579   0.114332
 0.1  -0.140303  0.108297   -0.0301525    -0.0605077   0.127905
 0.1  -0.136839  0.0975617  -0.0126751    -0.0783184   0.135994
 0.1  -0.133375  0.0870943   0.00354726   -0.0930919   0.139263
 0.1  -0.12991   0.0768952   0.0185465    -0.105025    0.138336
 ⋮                                                     ⋮
 0.1   0.133375  0.0870943  -0.00354726   -0.0930919  -0.139263
 0.1   0.136839  0.0975617   0.0126751    -0.0783184  -0.135994
 0.1   0.140303  0.108297    0.0301525    -0.0605077  -0.127905
 0.1   0.143767  0.119302    0.0489166    -0.0394579  -0.114332
 0.1   0.147232  0.130574    0.0689992    -0.014962   -0.0945669
 0.1   0.150696  0.142115    0.0904321     0.0131919  -0.0678636
 0.1   0.15416   0.153925    0.113247      0.0452209  -0.0334314
 0.1   0.157625  0.166002    0.137476      0.0813471   0.00956382
 0.1   0.161089  0.178349    0.16315       0.121798    0.0620008
 0.1   0.164553  0.190963    0.190302      0.166805    0.124804
 0.1   0.168017  0.203846    0.218963      0.216605    0.198945
 0.1   0.171482  0.216998    0.249164      0.271442    0.285443
```

In [11]:
```
using Gadfly
```

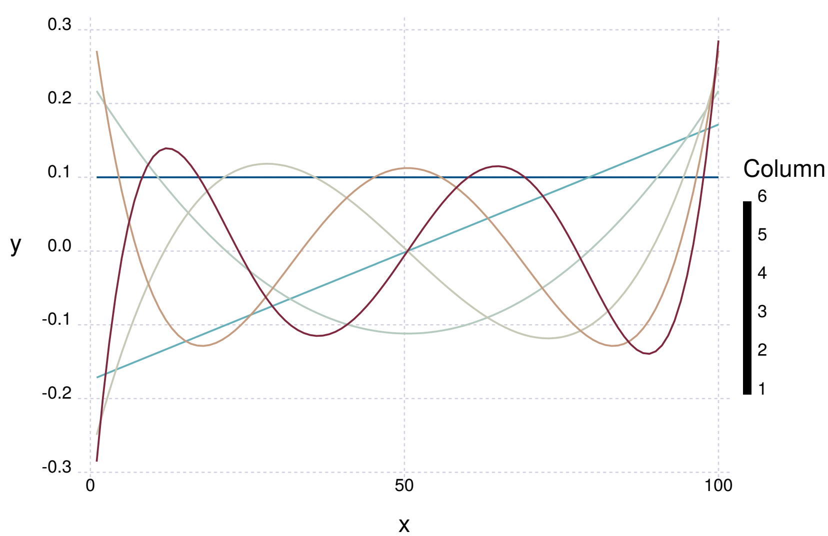
In [12]:
```
plot(V,x=Row.index, y=Col.value, color=Col.index, Geom.line)
```

Out[12]:



In [13]: plot(Q,x=Row.index, y=Col.value, color=Col.index, Geom.line)

Out[13]:



Dobiveni vektori su vrijednosti normiranih Legendreovih polinoma iz bilježnice NA12 Ortogonalni polinomi.ipynb.

Da bi dobili Čebiševljeve polinome, trebamo dodati težinsku funkciju ω i preraditi funkciju `myGramSchmidtQR()` iz bilježnice NA15 QR rastav.ipynb tako da računa *težinske skalarne produkte*. Dobiveni vektori su vrijednosti normiranih Čebiševljevih polinoma.

In [14]:
```
function myWeightedGramSchmidtQR(A::Array,ω::Vector)
    m,n=size(A)
    R=zeros(Float64,n,n)
    Q=Array{Float64,2}(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].*ω)/(Q[:,i]⋅(Q[:,i].*ω))
        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[14]:
```
myWeightedGramSchmidtQR (generic function with 1 method)
```

In [15]:
```
n=100
x=range(-0.99,stop=0.99,length=n)
ω=1 ./(sqrt.(1.0.-x.^2))
# Kvazi Vandermonde-ova matrica
V=[x.^0 x.^1 x.^2 x.^3 x.^4 x.^5]
```

Out[15]:
```
100×6 Array{Float64,2}:
 1.0  -0.99  0.9801  -0.970299  0.960596  -0.95099
 1.0  -0.97  0.9409  -0.912673  0.885293  -0.858734
 1.0  -0.95  0.9025  -0.857375  0.814506  -0.773781
 1.0  -0.93  0.8649  -0.804357  0.748052  -0.695688
 1.0  -0.91  0.8281  -0.753571  0.68575   -0.624032
 1.0  -0.89  0.7921  -0.704969  0.627422  -0.558406
 1.0  -0.87  0.7569  -0.658503  0.572898  -0.498421
 1.0  -0.85  0.7225  -0.614125  0.522006  -0.443705
 1.0  -0.83  0.6889  -0.571787  0.474583  -0.393904
 1.0  -0.81  0.6561  -0.531441  0.430467  -0.348678
 1.0  -0.79  0.6241  -0.493039  0.389501  -0.307706
 1.0  -0.77  0.5929  -0.456533  0.35153   -0.270678
 1.0  -0.75  0.5625  -0.421875  0.316406  -0.237305
 ⋮                                         ⋮
 1.0   0.77  0.5929   0.456533  0.35153    0.270678
 1.0   0.79  0.6241   0.493039  0.389501   0.307706
 1.0   0.81  0.6561   0.531441  0.430467   0.348678
```

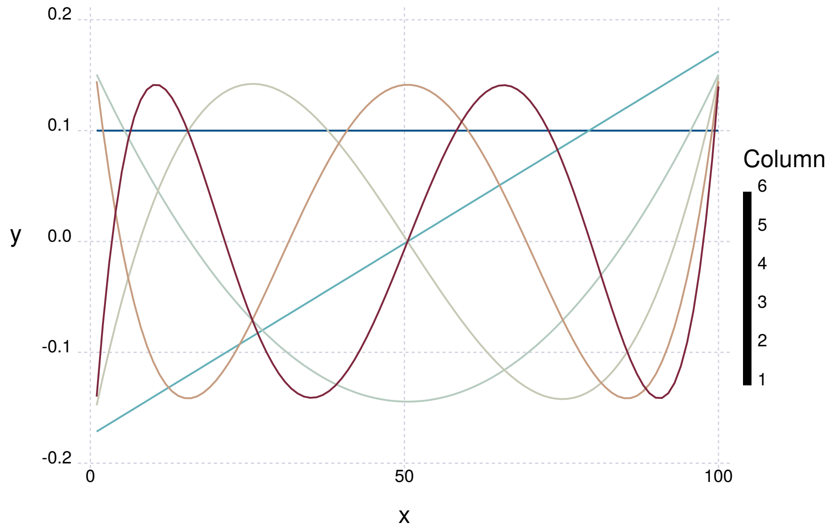
```
1.0   0.83  0.6889   0.571787  0.474583   0.393904
1.0   0.85  0.7225   0.614125  0.522006   0.443705
1.0   0.87  0.7569   0.658503  0.572898   0.498421
1.0   0.89  0.7921   0.704969  0.627422   0.558406
1.0   0.91  0.8281   0.753571  0.68575    0.624032
1.0   0.93  0.8649   0.804357  0.748052   0.695688
1.0   0.95  0.9025   0.857375  0.814506   0.773781
1.0   0.97  0.9409   0.912673  0.885293   0.858734
1.0   0.99  0.9801   0.970299  0.960596   0.95099
```

In [16]:
```
Q,R=myWeightedGramSchmidtQR(V,ω)
Q=Q*sign.(diagm(0=>diag(R)))
```

Out[16]:
```
100×6 Array{Float64,2}:
 0.1  -0.171482  0.150624   -0.147774     0.144547    -0.139908
 0.1  -0.168017  0.138822   -0.12221      0.100715    -0.0743526
 0.1  -0.164553  0.12726    -0.0980281    0.0613082   -0.0193541
 0.1  -0.161089  0.11594    -0.0752001    0.0261088    0.0259973
 0.1  -0.157625  0.10486    -0.0536975   -0.00509746   0.0625685
 0.1  -0.15416   0.0940209  -0.0334917   -0.0325198    0.091184
 0.1  -0.150696  0.0834228  -0.0145544   -0.056363     0.112627
 0.1  -0.147232  0.0730655   0.00314315  -0.0768273    0.127639
 0.1  -0.143767  0.0629492   0.0196293   -0.0941083    0.136924
 0.1  -0.140303  0.0530737   0.0349326   -0.108397     0.141146
 0.1  -0.136839  0.043439    0.0490815   -0.11988      0.14093
 0.1  -0.133375  0.0340453   0.0621045   -0.12874      0.136867
 0.1  -0.12991   0.0248924   0.0740302   -0.135153     0.12951
 ⋮                                                     ⋮
 0.1   0.133375  0.0340453  -0.0621045   -0.12874     -0.136867
 0.1   0.136839  0.043439   -0.0490815   -0.11988     -0.14093
 0.1   0.140303  0.0530737  -0.0349326   -0.108397    -0.141146
 0.1   0.143767  0.0629492  -0.0196293   -0.0941083   -0.136924
 0.1   0.147232  0.0730655  -0.00314315  -0.0768273   -0.127639
 0.1   0.150696  0.0834228   0.0145544   -0.056363    -0.112627
 0.1   0.15416   0.0940209   0.0334917   -0.0325198   -0.091184
 0.1   0.157625  0.10486     0.0536975   -0.00509746  -0.0625685
 0.1   0.161089  0.11594     0.0752001    0.0261088   -0.0259973
 0.1   0.164553  0.12726     0.0980281    0.0613082    0.0193541
 0.1   0.168017  0.138822    0.12221      0.100715     0.0743526
 0.1   0.171482  0.150624    0.147774     0.144547     0.139908
```

In [17]:
```
plot(Q,x=Row.index, y=Col.value, color=Col.index, Geom.line)
```

Out[17]:



0.2
0.1
0.0
-0.1
-0.2
y
0
50
100
x
Column
6
5
4
3
2
1