Constained

October 18, 2019

1 Constrained Optimization

Checking constrained least squares solution.

Generate a random 20 Œ 10 matrix A and a random 5Œ10 matrix C.

Then generate random vectors b and d of appropriate dimensions for the constrained least squares problem.

Compute the solution x.

1.0.1 solution 1: KKT Matrix

In [1]:
```
function cls_solve_kkt(A,b,C,d)
    m, n = size(A)
    p, n = size(C)
    G = A'*A # Gram matrix
    KKT = [2*G C'; C zeros(p,p)] # KKT matrix
    xzhat = KKT \ [2*A'*b; d]
    return xzhat[1:n,:]
end
```

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

1.0.2 solution 2: QR Factorization

In [2]:
```
using LinearAlgebra
function cls_solve(A,b,C,d)
    m, n = size(A)
    p, n = size(C)
    Q, R = qr([A; C])
    Q = Matrix(Q)
    Q1 = Q[1:m,:]
    Q2 = Q[m+1:m+p,:]
    Qtil, Rtil = qr(Q2')
    Qtil = Matrix(Qtil)
    w = Rtil \ (2*Qtil'*Q1'*b - 2*(Rtil'\d))
    return xhat = R \ (Q1'*b - Q2'*w/2)
end
```

Out[2]:
```
cls_solve (generic function with 1 method)
```

In [3]:
```
m = 20; n = 10; p = 5;
A = randn(m,n); b = randn(m); C = randn(p,n); d = randn(p);
# The result of QR method
xQR = cls_solve(A,b,C,d)
```

Out[3]:
```
10-element Array{Float64,1}:
 -0.9349161762700104
  0.13579039800470588
  0.1655293673895859
  0.11233994304786578
 -0.218401904898281
  0.4091444232458048
 -0.7173668727795459
  0.24892271901771604
 -0.286204336498991
  0.2855729688947593
```

In [4]:
```
# The result of KKT method
xKKT = cls_solve_kkt(A,b,C,d)
```

Out[4]:
```
10Œ1 Array{Float64,2}:
 -0.9349161762700114
  0.13579039800470646
  0.1655293673895863
  0.1123399430478662
 -0.21840190489828082
  0.4091444232458051
 -0.7173668727795468
  0.24892271901771654
 -0.2862043364989908
  0.2855729688947593
```

1.0.3 compare the results of two methods

In [5]:
```
norm(xKKT-xQR)
```

Out[5]:
```
1.6926350419598586e-15
```