In [43]:

```
using Random, LinearAlgebra, Statistics, StatsPlots
```

In [44]:

```
# JuMP is a front-end for optimization.  The rest are backends.
using JuMP, SCS, GLPK, Mosek, MosekTools
```

In [45]:

```
# These simplify choice of backends with the parameters I want.
opt_scs = optimizer_with_attributes(SCS.Optimizer, "verbose"=>0)
opt_glpk = optimizer_with_attributes(GLPK.Optimizer, "msg_lev" =
> GLPK.OFF)
opt_mosek = optimizer_with_attributes(Mosek.Optimizer, "QUIET" =
> true)
```

Out[45]:

```
MathOptInterface.OptimizerWithAttributes(Mosek.Optim
izer, Pair{MathOptInterface.AbstractOptimizerAttribu
te,Any}[MathOptInterface.RawParameter("QUIET") => tr
ue])
```

In [46]:

```
onenorm(x) = norm(x,1)
onenorm([1 2 3])
```

Out[46]:

```
6.0
```

Incoherence

In [47]:

```
d = 5
n = 20
Random.seed!(0)
A = randn(d,n)
```

Out[47]:

```
5×20 Array{Float64,2}:
  0.679107   0.297336   -0.688907  …   0.556741   0.
441573   1.3378
  0.828413   0.0649475  -0.762804     -0.568854   1.
67689    1.21963
 -0.353007  -0.109017    0.397482     -0.47694    0.
238481  -1.22187
 -0.134854  -0.51421     0.81163       0.174594   2.
31379    0.256914
  0.586617   1.57433    -0.346355     -1.51761   -0.
081851   0.799711
```

Compute Incoherence

In [48]:

```
normed_A = similar(A)
for i in 1:n
    normed_A[:,i] = A[:,i] / norm(A[:,i])
end
Gram = normed_A' * normed_A - I
```

```
Out[48]:

20×20 Array{Float64,2}:
  2.22045e-16   0.596628   -0.860283      …   0.33547
8       0.930847
  0.596628      0.0        -0.52904          -0.2255
0.439922
 -0.860283     -0.52904    -2.22045e-16       0.10194
6      -0.728318
 -0.159119     -0.113124    0.405184          0.17301
6       0.180499
 -0.473822      0.128045    0.439549         -0.26327
5      -0.670492
 -0.672912     -0.813001    0.620785      …   0.10799
3      -0.587188
  0.684779      0.546908   -0.618257          0.13914
2       0.523754
  0.425619      0.363483   -0.0511253         0.40163
3       0.654449
 -0.420792     -0.784395    0.165879         -0.11962
8      -0.337706
  0.734215      0.413084   -0.579377          0.43877
2       0.58571
 -0.283173     -0.0166007   0.0309163     …  -0.67892
2      -0.11669
 -0.342546     -0.624092    0.140511         -0.23060
8      -0.0875969
 -0.622748     -0.467635    0.16071          -0.74733
5      -0.596043
 -0.55152       0.0298985   0.740323         -0.11047
2      -0.37182
  0.313471      0.643136   -0.214969          0.01017
17      0.101295
 -0.292367     -0.907657    0.155125      …   0.23793
7      -0.190913
 -0.564701     -0.783707    0.643148          0.42311
7      -0.558533
 -0.366999     -0.762058    0.2093           -0.05659
86     -0.12792
  0.335478     -0.2255      0.101946         -1.11022
e-16    0.423089
  0.930847      0.439922   -0.728318          0.42308
9       0.0
```

In [49]:

```
μ = maximum(abs.(Gram))
```

Out[49]:

```
0.9679382785673595
```

In [50]:

```
function incoherence(A)
    normed_A = similar(A)
    for i in 1:size(A,2)
        normed_A[:,i] = A[:,i] / norm(A[:,i])
    end
    Gram = normed_A' * normed_A - I
    return maximum(abs.(Gram))
end
```

Out[50]:

```
incoherence (generic function with 1 method)
```

In [57]:

```
A = randn(5*25,5*100)
incoherence(A)
```

Out[57]:

```
0.4155190340602961
```

Basis Pursuit by JuMP

In [58]:

```
d = 10
n = 20
A = randn(d,n)
x0 = zeros(n)
x0[1] = 1.0
b = A*x0
```

Out[58]:

```
10-element Array{Float64,1}:
 -0.11248681351342672
 -0.5662892741372261
 -0.14948376404232092
 -0.7541284390976164
 -0.35188939912748723
  1.7002809122995806
 -1.9143586614424437
  0.310830838565587
  0.3351183436255904
  0.5234120006014658
```

In [59]:

```
model = Model(SCS.Optimizer)
```

Out[59]:

feasibility
Subject to

In [60]:

```
@variable(model, x[1:n])
```

Out[60]:

```
20-element Array{VariableRef,1}:
 x[1]
 x[2]
 x[3]
 x[4]
 x[5]
 x[6]
 x[7]
 x[8]
 x[9]
 x[10]
 x[11]
 x[12]
 x[13]
 x[14]
 x[15]
 x[16]
 x[17]
 x[18]
 x[19]
 x[20]
```

In [61]:

```
@constraint(model, A*x .== b)
```

Out[61]:

```
10-element Array{ConstraintRef{Model,MathOptInterfac
e.ConstraintIndex{MathOptInterface.ScalarAffineFunct
ion{Float64},MathOptInterface.EqualTo{Float64}},Scal
arShape},1}:
 -0.11248681351342672 x[1] - 1.2317225437811226 x[2]
+ 0.60532364592491 x[3] + 0.612402873744б184 x[4] +
0.6528565097491319 x[5] + 1.6979580898265894 x[6] -
0.6862486265217821 x[7] + 0.15289903450089515 x[8] -
0.004772539509995808 x[9] + 0.9716086943526653 x[10]
- 0.9042366324633847 x[11] - 0.6269118217601528 x[12
] + 0.8970043567906548 x[13] - 0.4667104302119095 x[
14] + 0.3817729672243532 x[15] - 0.19991603062192095

x[16] + 0.6758885311998514 x[17] + 0.441577266009788
25 x[18] + 0.9334671338170232 x[19] - 0.523082339678
034 x[20] = -0.11248681351342672
 -0.5662892741372261 x[1] + 0.5674902277653299 x[2]
+ 1.1163665397172606 x[3] + 0.4567725810041199 x[4]
+ 1.357259294102547 x[5] + 1.1523743380963307 x[6] -
0.11599185003107795 x[7] + 3.1476344298475682 x[8] +
1.1042089249028642 x[9] - 0.16524890818251917 x[10]
- 0.9086133913056114 x[11] - 0.22708133243023448 x[1
2] - 1.119918619955327 x[13] - 0.6035039380546332 x[
14] + 0.26272948667516594 x[15] - 1.0513935582164728
x[16] + 0.44570087758409865 x[17] + 0.59569578570560
84 x[18] + 0.5821841889434244 x[19] - 0.198891383154
38443 x[20] = -0.5662892741372261
 -0.14948376404232092 x[1] + 1.5517163112180166 x[2]
- 1.7256775636583113 x[3] - 1.4001529409440712 x[4]
- 0.40816543905259284 x[5] + 1.2256635223911487 x[6]
- 0.9095984379305072 x[7] + 0.8158670916316041 x[8]
- 0.33253786352787884 x[9] - 1.6199664190518903 x[10
] + 2.2825810094694656 x[11] + 0.337284707387779 x[1
2] + 0.7416188228601653 x[13] - 0.6498398772782128 x
[14] + 1.675344752615471 x[15] - 0.9999991635264687
x[16] + 0.2458102783782662 x[17] + 0.850114386589317
x[18] + 0.8493338819701169 x[19] - 1.095166371145091
x[20] = -0.14948376404232092
 -0.7541284390976164 x[1] - 1.4941525933748385 x[2]
+ 1.0832040669018208 x[3] - 0.15912656304085246 x[4]
+ 1.751498033558951 x[5] + 0.4864199247342999 x[6] +
1.023917356144l013 x[7] + 0.06718640764610144 x[8] +
0.8900716414239433 x[9] + 0.5749389317056772 x[10] -
0.863852432912855l x[11] + 1.792164501718729 x[12] +
0.3343406337434719 x[13] + 2.235498627467402 x[14] -
1.777617585423880З x[15] - 1.0258963869354887 x[16]
- 0.492852377l749276 x[17] + 0.3952410710252834 4 x[1
8] - 1.0638415363597291 x[19] + 0.4428400393578505 x
[20] = -0.7541284390976164
 -0.3518893991274872З x[1] - 0.556451676529748 7 x[2]
+ 0.8609077087689985 x[3] - 0.4459580353344765 x[4]
- 1.3787248167701522 x[5] - 0.524354973448415 x[6] -
0.3080951372473763 6 x[7] + 0.4870258380831458 x[8] -
1.167992173773309 9 x[9] - 0.2544292702091051 6 x[10]
+ 0.566665294422920 7 x[11] - 1.0221727823880569 x[12
] - 1.0936761381380715 x[13] + 0.4953375902470542 x[
14] + 0.7614509551498848 x[15] + 0.3488858367575624
x[16] + 0.01227993536585 4578 x[17] + 1.2274476167788
15 x[18] - 1.0170144870456475 x[19] + 0.617309905365
5823 x[20] = -0.3518893991274 8723
 1.7002809122995806 x[1] - 1.3080362019937761 x[2] +
0.6698914942502485 x[3] - 1.1474334649183995 x[4] +
0.1105052625525060 9 x[5] + 0.8995290099838263 x[6] -
0.663447810553044 5 x[7] - 0.0523541230546904 7 x[8] -
0.0298851410002027 6 x[9] + 1.1521383853656486 x[10]
- 0.0606330172970474 3 x[11] + 0.8365335929703048 x[1
2] - 0.3122393686558382 3 x[13] + 0.0260250054916220 2
x[14] - 0.0357162472001147 65 x[15] + 0.5853579238128
782 x[16] - 0.6052857888172622 x[17] - 0.62935204975
36604 x[18] - 0.4989225029374304 x[19] + 0.000741189
6423990069 x[20] = 1.7002809122995806
 -1.9143586614424437 x[1] + 0.2332720816757861 x[2]
- 0.9108041941774613 x[3] + 0.6718497718092494 x[4]
- 0.3864706275744775 x[5] + 0.9430354905454666 x[6]
- 1.9040919552655386 x[7] + 1.9572258747799127 x[8]
- 0.1871357685392489 6 x[9] - 1.6269066173163877 x[10
] + 0.2841854095854801 6 x[11] - 0.0469974422829074 6
x[12] + 1.2886659385255383 x[13] + 0.351322509466695
5 x[14] + 0.5512684747303959 x[15] - 0.0889196061265
8953 x[16] - 1.8808964982943022 x[17] - 0.2376802282
037522 x[18] - 0.6731284617927262 x[19] - 1.21586630
83813768 x[20] = -1.9143586614424437
 0.310830838565587 x[1] + 0.8888270096032808 x[2] +
0.7577760165487661 x[3] - 0.1551888535910063 1 x[4] -
0.768484471400180 8 x[5] + 0.7497799172643206 x[6] -
0.3506279091819655 7 x[7] - 2.6337340518702805 x[8] -
1.4851587993581619 x[9] - 0.4348010860765001 4 x[10]
- 0.3557358498359327 6 x[11] + 0.4466884954755695 x[1
2] + 0.0687920048035756 1 x[13] + 0.3449030436283199 7
x[14] + 0.2245176084665537 8 x[15] - 1.13399888011798
88 x[16] - 0.0839054398350979 6 x[17] - 0.36613661255
64423 x[18] - 1.125539882856653 x[19] + 0.5498958158
84775 x[20] = 0.310830838565587
 0.3351183436255904 x[1] - 0.2022402952776496 4 x[2]
+ 0.3728775028304399 7 x[3] + 0.9197793389347283 x[4]
- 2.3555443079883993 x[5] + 1.2171416584700034 x[6]
+ 1.06385210260751 3 x[7] - 0.6863502981597671 x[8] +
1.3033516262799636 x[9] + 0.3078625747412886 4 x[10]
+ 0.1556018049617877 7 x[11] - 1.2368309277212384 x[1
2] - 0.964556585460038 5 x[13] + 0.2816474936831833 x
[14] + 0.6622404850086145 x[15] - 1.3613390912274006
```

```
x[16] - 0.323996338035808 x[17] + 0.5074502141260505
x[18] - 0.1329498331538597 x[19] - 0.290526579329645
93 x[20] = 0.3351183436255904
 0.5234120006014658 x[1] + 0.15127595075684833 x[2]
+ 1.9741941124149873 x[3] - 2.3005881197896825 x[4]
+ 0.6298493295838168 x[5] - 2.3097456337123443 x[6]
+ 1.927549279609831 x[7] + 0.4347452916044357 x[8] -
0.05169266614265093 x[9] - 1.179174687395125 x[10] -
2.0405758671455123 x[11] - 0.027821906466193414 x[12
] - 0.8486853210219902 x[13] + 0.13773693582241403 x
[14] - 0.8753206403726591 x[15] + 0.4334144798605275
6 x[16] + 0.13045667892447826 x[17] + 0.935996643261
1737 x[18] + 0.649770818201045 x[19] + 1.29416949659
06838 x[20] = 0.5234120006014658
```

In [62]:

```
@variable(model, y[1:n])
@constraint(model, x .<= y)
@constraint(model, -x .<= y)
@objective(model, Min, sum(y))
```

Out[62]:

$$y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 + y_9 + y_{10} + y_{11} + y_{12} +$$

In [63]:

```
optimize!(model)
```

```
----------------------------------------------------
------------------------
        SCS v2.1.1 - Splitting Conic Solver
        (c) Brendan O'Donoghue, Stanford University,
2012
----------------------------------------------------
------------------------
Lin-sys: sparse-indirect, nnz in A = 280, CG tol ~ 1
/iter^(2.00)
eps = 1.00e-05, alpha = 1.50, max_iters = 5000, norm
alize = 1, scale = 1.00
acceleration_lookback = 10, rho_x = 1.00e-03
Variables n = 40, constraints m = 50
Cones:  primal zero / dual free vars: 10
        linear vars: 40
```

```
Setup time: 1.03e-04s
----------------------------------------------------
------------------------
 Iter | pri res | dua res | rel gap | pri obj | dua
obj | kap/tau | time (s)
----------------------------------------------------
------------------------
     0| 3.74e+19  6.44e+19  1.00e+00 -2.10e+20  1.43
e+20  9.17e+19  7.25e-05
    40| 1.62e-07  8.12e-07  4.43e-07  1.00e+00  1.00
e+00  2.06e-16  1.32e-03
----------------------------------------------------
------------------------
Status: Solved
Timing: Solve time: 1.32e-03s
        Lin-sys: avg # CG iterations: 2.76, avg solv
e time: 1.54e-05s
        Cones: avg projection time: 1.45e-07s
        Acceleration: avg step time: 1.32e-05s
----------------------------------------------------
------------------------
Error metrics:
dist(s, K) = 4.2368e-17, dist(y, K*) = 0.0000e+00, s
'y/|s||y| = 5.8604e-18
primal res: |Ax + s - b|_2 / (1 + |b|_2) = 1.6239e-0
7
dual res:   |A'y + c|_2 / (1 + |c|_2) = 8.1246e-07
rel gap:    |c'x + b'y| / (1 + |c'x| + |b'y|) = 4.42
89e-07
----------------------------------------------------
------------------------
c'x = 1.0000, -b'y = 1.0000
====================================================
========================
```

In [64]:

```
xs = value.(x)
```

Out[64]:

```
20-element Array{Float64,1}:
  1.0000002097854823
 -3.719467630711012 5e-8
  1.3057975606922636e-8
 -6.652681661529527e-9
  2.0993775947052414e-8
 -6.769918215855081e-9
 -2.06070404672574e-8
 -2.1598410376842593e-8
  6.107184897131848e-9
 -3.68665891009158e-8
  3.4011270251743704e-8
  9.123559643923159e-10
 -2.31421917475553e-8
 -5.048228113906143e-8
  8.591719552765204e-9
 -1.3121841985150527e-8
 -5.433844722264766e-8
  1.504538540165436e-8
  2.625446251349709e-8
 -5.771370350149791e-8
```

In [65]:

```
onenorm(xs - x0)
```

Out[65]:

```
6.632473941796915e-7
```

Let's make a function out of that.

In [ ]:

```
function BP(A,b; optimizer = opt_scs)

    d, n = size(A)
    @assert length(b) == d

    model = Model(optimizer)
    @variable(model, x[1:n])
    @constraint(model, A*x .== b)

    @variable(model, y[1:n])
    @constraint(model, x .<= y)
    @constraint(model, -x .<= y)
    @objective(model, Min, sum(y))

    optimize!(model)

    xs = value.(x)

    return xs
end
```

In [66]:

```
xs = BP(A,b, optimizer=opt_glpk)
```

Out[66]:

```
20-element Array{Float64,1}:
 1.0
 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
```

In [67]:

```
onenorm(xs-x0)
```

Out[67]:

```
0.0
```

In [68]:

```
xs = BP(A,b, optimizer=opt_mosek)
onenorm(xs-x0)
```

Out[68]:

```
3.5449407029009996e-17
```

In [69]:

```
randset(n,k) = randperm(n)[1:k]
randset(10,3)
```

Out[69]:

```
3-element Array{Int64,1}:
 1
 3
 6
```

In [70]:

```
d = 50
n = 200
k = 7
S = randset(n,k)
x0 = zeros(n)
x0[S] = randn(k)
x0
```

```
Out[70]:

200-element Array{Float64,1}:
  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
 -1.0444736865400661
  0.0
  0.0
  0.0
  0.0
  0.0
  0.0
  0.0
  0.0
  0.0
  0.0
```

In [71]:

```
sparse(x0)
```

Out[71]:

```
200-element SparseVector{Float64,Int64} with 7 store
d entries:
  [29 ]  =  -0.285493
  [38 ]  =  -0.599437
  [87 ]  =  -1.27762
  [107]  =  -0.580424
  [123]  =  0.915584
  [143]  =  0.321277
  [190]  =  -1.04447
```

In [72]:

```
A = randn(d,n)
b = A*x0
xs = BP(A,b)
onenorm(x0-xs)
```

Out[72]:

```
1.3954734856354438e-5
```

In [73]:

```
function make_problem(d,n,k)
    S = randset(n,k)
    x0 = zeros(n)
    x0[S] = randn(k)
    A = randn(d,n)
    b = A*x0;
    return A,b,x0
end
A, b, x0 = make_problem(d, n, k)
```

Out[73]:

```
([1.4302075533475465 0.7520118815077361 … -1.4701788
233252002 -0.35555099557525566; 0.7812316922601905 0
.19914232040424418 … -0.9954739771581482 0.264203759
74891974; … ; 0.7734909444638944 -0.6865218343782568
… -0.19714239408002335 -0.3838898158942502; 0.864779
2696114751 1.0643908284555994 … 1.4109178925704224 0
.8426275793521971], [1.7650602445585468, 4.605709316
9563865, -1.7296368096215045, -0.19432879001198017,
0.9264485353718146, -0.1862857479190762, 1.040562946
1544077, -1.1371974562444191, -0.3326791218206496, -
6.230899766426076  …  -2.734486046085591, -2.6126636
8229103, -3.489783791060211, 5.010377527588336, -0.7
31604765215425, -2.674764514776432, -1.9619302473929
523, 5.069184704505968, -0.9854364907820127, 1.02453
33091398552], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 1.194546502452986, 0.0  …  0.0, 0.0, 0.0, 0.0, 0
.0, 0.0, 0.0, 0.0, 0.0, 0.0])
```

In [74]:

```
A, b, x0 = make_problem(50, 200, 15)
xs = BP(A,b)
onenorm(x0-xs)
```

Out[74]:

```
0.3553601360338345
```

In [75]:

```
onenorm(x0)
```

Out[75]:

```
9.25872226678312
```

In [76]:

```
@time xs = BP(A,b,optimizer=opt_scs)
@time xs = BP(A,b,optimizer=opt_glpk)
@time xs = BP(A,b,optimizer=opt_mosek)
;
```

```
  5.881336 seconds (61.02 k allocations: 9.716 MiB)
  0.048013 seconds (30.03 k allocations: 3.071 MiB)
  0.066250 seconds (40.00 k allocations: 4.095 MiB)
```

In [77]:

```
A, b, x0 = make_problem(200, 500, 100)
@time x1 = BP(A,b,optimizer=opt_glpk)
@time x2 = BP(A,b,optimizer=opt_mosek)
onenorm(x1-x0), onenorm(x2-x0)
```

```
  1.576325 seconds (79.33 k allocations: 16.935 MiB,
1.69% gc time)
  0.660002 seconds (104.77 k allocations: 19.672 MiB
)
```

Out[77]:

```
(42.337713825009715, 42.33771382500943)
```

In [78]:

```
A, b, x0 = make_problem(200, 500, 40)
@time x1 = BP(A,b,optimizer=opt_glpk)
@time x2 = BP(A,b,optimizer=opt_mosek)
onenorm(x1-x0), onenorm(x2-x0)
```

```
  1.098129 seconds (79.33 k allocations: 16.935 MiB)
  0.914337 seconds (104.77 k allocations: 19.700 MiB
)
```

Out[78]:

```
(6.133142236098867e-13, 1.1976518734336954e-13)
```

BPDN

In [ ]:

```
function make_problem(d,n,k,ϵ)
    S = randset(n,k)
    x0 = zeros(n)
    x0[S] = randn(k)

    A = randn(d,n)
    for i in 1:n
        A[:,i] = A[:,i] / norm(A[:,i])
    end

    e = randn(d)
    e = ϵ * e / norm(e)
    b = A*x0 + e

    return A, b, x0
end
```

In [79]:

```
A, b, x0 = make_problem(10,50,2,0.1)
```

Out[79]:

```
([-0.31455647227518035 -0.008090775210207369 … -0.17
042816394751953 -0.19542786176733629; -0.05731604790
835892 0.1868328526942932 … -0.06236958591823509 -0.
34149074468279744; … ; 0.20562128249380562 -0.412134
9455984317 … 0.14460064093060726 0.1688491475418608;
0.3811648015138O894 0.4977991652864089 … 0.097104060
80012117 0.11245830487696122], [-0.9056498409818027,
-1.400990935048949, 0.18839297043983338, -0.05102212
802650892, -0.8472918886171541, -0.0379989896072651 2
, 0.08314299597279266, -0.4346907602594959, -0.34503
047689841976, 0.6672663786293269], [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, 0.754102336582079, 0.0, 0.0])
```

In [80]:

```
function BPDN(A,b,ϵ; optimizer = opt_scs)

    d, n = size(A)
    @assert length(b) == d

    model = Model(optimizer)
    @variable(model, x[1:n])
    @constraint(model, [ϵ; A*x - b] in SecondOrderCone()) # norm
(Ax - b) ≤ ϵ

    @variable(model, y[1:n])
    @constraint(model, x .<= y)
    @constraint(model, -x .<= y)
    @objective(model, Min, sum(y))

    optimize!(model)

    xs = value.(x)

    return xs
end
```

Out[80]:

```
BPDN (generic function with 1 method)
```

In [81]:

```
xs = BPDN(A,b,0.1)
norm(xs-x0), onenorm(xs-x0)
```

Out[81]:

```
(0.114430905146159, 0.2304691248833006)
```

In [82]:

```
norm(x0), norm(xs)
```

Out[82]:

```
(2.1189445683491557, 2.025754594527242)
```

In [83]:

```
A, b, x0 = make_problem(10,50,2,0.01)
xs = BPDN(A,b,0.01)
norm(xs-x0), norm(x0)
```

Out[83]:

```
(0.01502772364275557, 1.2763608041825145)
```

In [84]:

```
A, b, x0 = make_problem(100,500,20,0.1)
xs = BPDN(A,b,0.01, optimizer=opt_glpk)
norm(xs-x0), norm(x0)
```

```
Constraints of type MathOptInterface.VectorAffineFun
ction{Float64}-in-MathOptInterface.SecondOrderCone a
re not supported by the solver and there are no brid
ges that can reformulate it into supported constrain
ts.

Stacktrace:
 [1] error(::String) at ./error.jl:33
 [2] moi_add_constraint(::MathOptInterface.Utilities
.CachingOptimizer{MathOptInterface.AbstractOptimizer
,MathOptInterface.Utilities.UniversalFallback{MathOp
tInterface.Utilities.Model{Float64}}}, ::MathOptInte
rface.VectorAffineFunction{Float64}, ::MathOptInterf
ace.SecondOrderCone) at /Users/spielman/.julia/packa
ges/JuMP/MnJQc/src/constraints.jl:389
 [3] add_constraint(::Model, ::VectorConstraint{Gene
ricAffExpr{Float64,VariableRef},MathOptInterface.Sec
ondOrderCone,VectorShape}, ::String) at /Users/spiel
man/.julia/packages/JuMP/MnJQc/src/constraints.jl:40
3
 [4] macro expansion at /Users/spielman/.julia/packa
ges/JuMP/MnJQc/src/macros.jl:382 [inlined]
 [5] #BPDN#27(::MathOptInterface.OptimizerWithAttrib
utes, ::typeof(BPDN), ::Array{Float64,2}, ::Array{Fl
oat64,1}, ::Float64) at ./In[80]:8
 [6] (::var"#kw##BPDN")(::NamedTuple{(:optimizer,),T
uple{MathOptInterface.OptimizerWithAttributes}}, ::t
ypeof(BPDN), ::Array{Float64,2}, ::Array{Float64,1},
::Float64) at ./none:0
 [7] top-level scope at In[84]:2
```

In [85]:

```
A, b, x0 = make_problem(100,500,20,0.1)
xs = BPDN(A,b,0.01, optimizer=opt_mosek)
norm(xs-x0), norm(x0)
```

Out[85]:

```
(0.34580576413952496, 3.77540385217923)
```

In [86]:

```
function plot_compare(x0, xs, k)
    @assert any(x0 .== 0) # make sure got order of x0 vs xs corr
ect
    p = sortperm(abs.(xs) + abs.(x0),rev=true)
    groupedbar(hcat(x0[p[1:k]] , xs[p[1:k]]), label=hcat("x0", "
xs"))
end
```

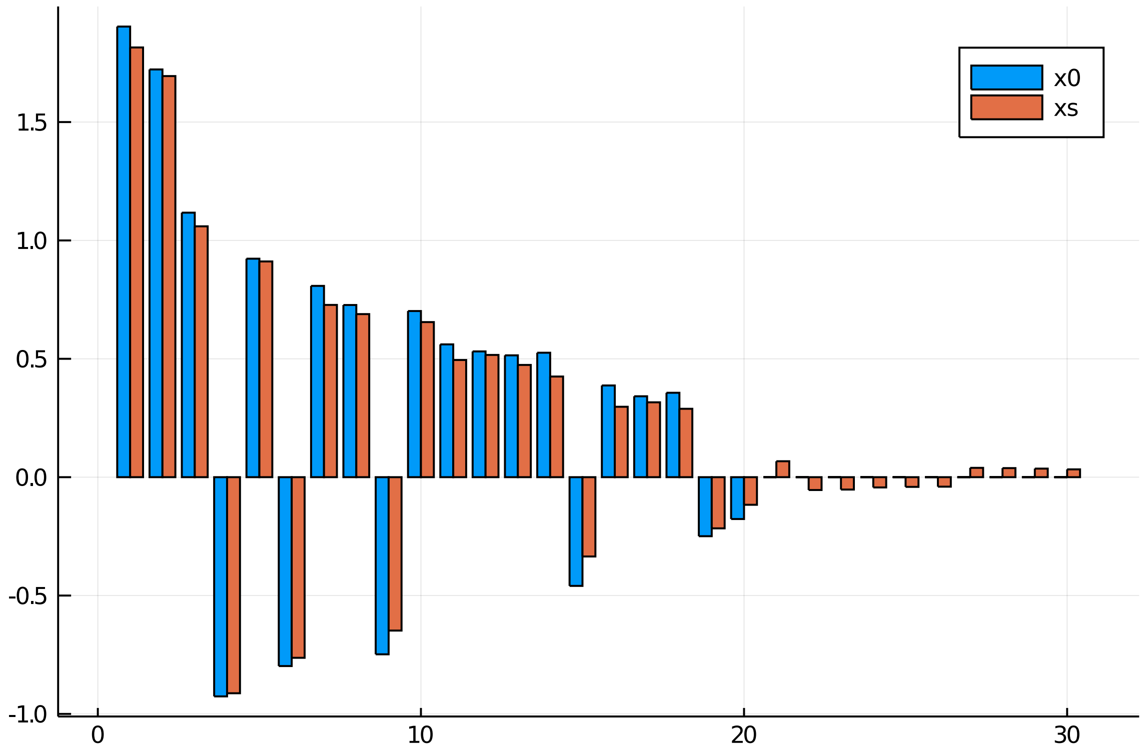
Out[86]:

```
plot_compare (generic function with 1 method)
```

In [87]:

```
plot_compare(x0, xs, 30)
```

Out[87]:

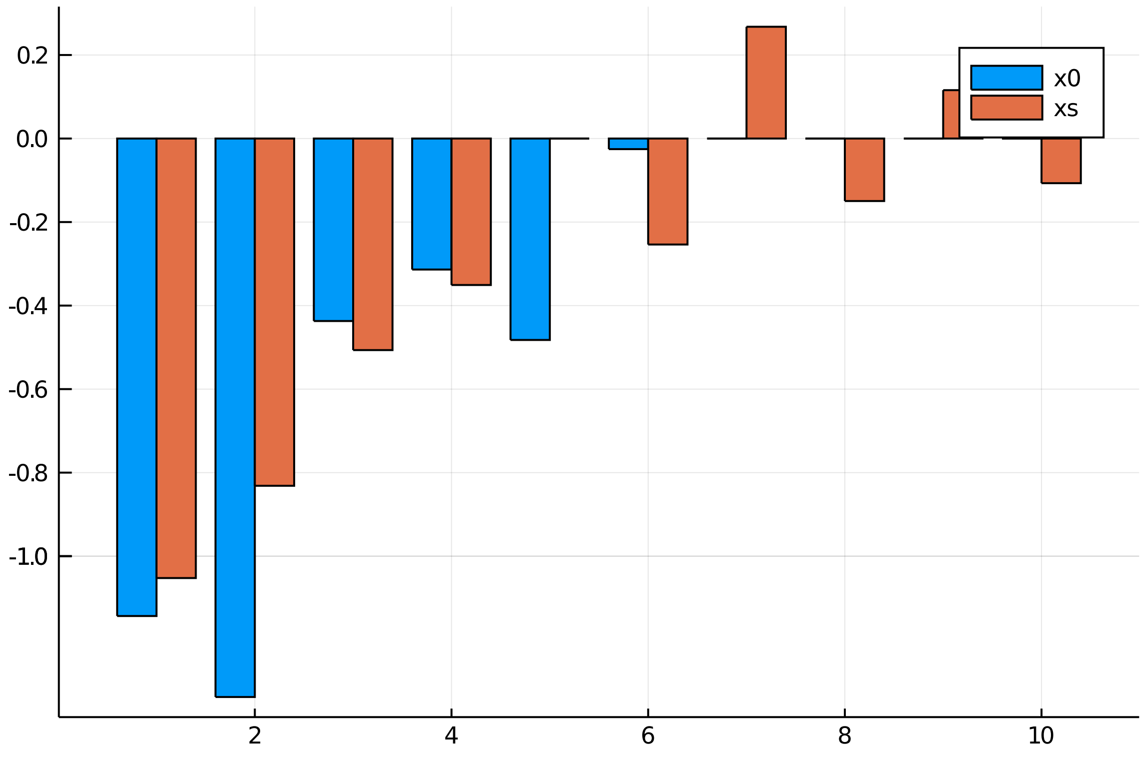


In [88]:

```
A, b, x0 = make_problem(10, 30, 6)
xs = BP(A,b)
@show norm(xs-x0), norm(x0)
plot_compare(x0, xs, 10)
```

```
(norm(xs - x0), norm(x0)) = (0.8268405421419495, 1.9
021607993177583)
```

Out[88]:



In [90]:

```
d = 200
n = 400
k = 50
A, b = make_problem(d, n, k)
@time xs = BP(A,b, optimizer=opt_mosek)

A, b, x0 = make_problem(d, n, k, 0.1)
@time xs = BPDN(A,b, 0.1, optimizer=opt_mosek)
;
```

```
  0.449484 seconds (87.56 k allocations: 17.339 MiB)
  0.276066 seconds (87.81 k allocations: 28.340 MiB,
8.63% gc time)
```

In [ ]:

```
norm(xs-x0), norm(x0)
```

In [ ]:

```

```