code

January 11, 2018

In [1]:
```
using PyPlot
=1
=3
iteration=10^4
N=10
M=N
```

Out[1]: 10

In [2]:
```
# compute, to how big error step with omega size would lead
function errorOmega(x, xavr, xopt, omega)
    return norm( ((1-omega)*x + omega*xavr) - xopt )
end
```

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

In [3]:
```
# compute size of the best  for particular point x
# starts wih interval, where to look for , splits it into 3 parts, throw away wrong part
# suppose, that error is unimodal function
function getBestOmega(x, xavr, xopt, lower, upper, accuracy)
    while upper-lower>accuracy
        middle1=(2*lower+upper)/3
        middle2=(lower+2*upper)/3

        if errorOmega(x, xavr, xopt, middle1) > errorOmega(x, xavr, xopt, middle2)
            lower=middle1
        else
            upper=middle2
        end
    end
    return (lower+upper)/2
end
```

Out[3]: getBestOmega (generic function with 1 method)

In [4]:
```
# get a random vector of size M+1 with  ones
function setS()
    S=zeros(M+1,1)
```

```
    for i=1:
        r=rand(1:M+1)
        while S[r]==1
            r=rand(1:M+1)
        end
        S[r]=1
    end
    return S
end
```

Out[4]: setS (generic function with 1 method)

In [5]:
```
# compute average of projections
function projection(A, b, x)
    S=setS()
    xsum=x*0
    for coord=1:M
        if S[coord]==1
            xsum += x - A[coord,:].*(A[coord,:]'*x - b[coord])/(A[coord,:]A[coord,:])
        end
    end
    if S[M+1]==1
        # project on ball
        if norm(x)>10
            xsum += 10*x/norm(x)
        else
            xsum += x
        end
    end
    return xsum/
end
```

Out[5]: projection (generic function with 1 method)

In [6]:
```
# randomly initialize variables
A=randn(M,N)
b=randn(M)
x0=1000*rand(M)
println("norm xopt: ", norm(A\b))
println("norm x0: ", norm(x0))
```

```
norm xopt: 2.567477664141803
norm x0: 1784.3656339772504
```

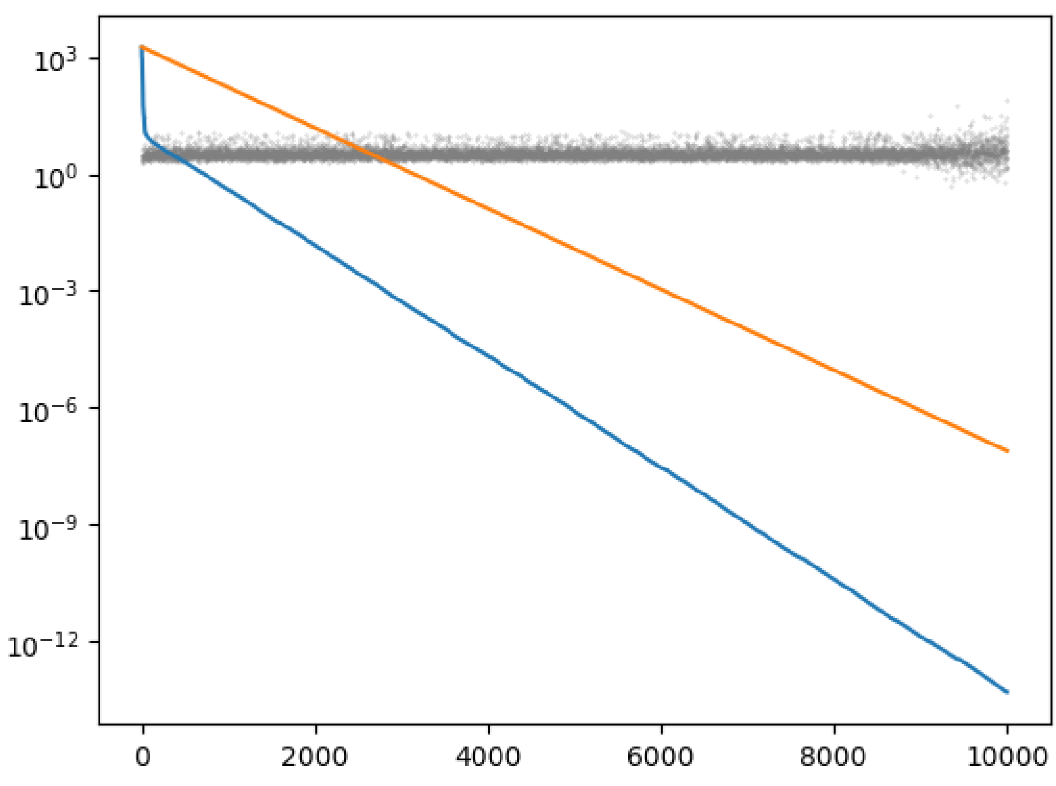
In [7]:
```
# find projection to measure distances (same algorithm runned 10x longer)
xopt=x0
for t=1:10*iteration
    xproj = projection(A,b,xopt)
```

```
    xopt = (1-)*xopt + *xproj
end
print("xopt:", norm(A*xopt-b))
```

```
xopt:2.9748540179341104e-15
```

In [8]:
```
# run algorithm and save variables to be plotted
dist=zeros(iteration)
bestOmega=zeros(iteration)
x=x0
for t=1:iteration
    dist[t] = norm(x-xopt)
    xproj = projection(A,b,x)
    bestOmega[t] = getBestOmega(x, xproj, xopt, 0, 1000, 1/10^10)
    x = (1-)*x + *xproj # project on
    # x = (1-bestOmega[t])*x + bestOmega[t]*xproj # project on best omega
end
```

In [9]:
```
# plot variables
semilogy(dist)
v = eigvals(A'*A)
min = minimum(v[v.>10.0^(-13)]) # for us it is not zero, but for comp it is
rate = 1 - min/sum(v)
semilogy(1:iteration,dist[1]*rate.^(1:iteration))
scatter(1:iteration,bestOmega, s=0.1, color="grey", alpha=0.5)
```



```
Out[9]: PyObject <matplotlib.collections.PathCollection object at 0x1228dfd90>
```