```
using PyPlot
omega=1
tau=3
iteration=10^4
N=10
M=N

# compute, to how big error step with omega size would lead
function errorOmega(x, xavr, xopt, local_omega)
   return norm( ((1-local_omega)*x + local_omega*xavr) - xopt )
end

# compute size of the best omega for particular point x
# starts wih interval, where to look for omega, 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

# get a random vector of size M+1 with tau ones
function setS()
   S=zeros(M+1,1)
 for i=1:tau
  r=rand(1:M+1)
  while S[r]==1
   r=rand(1:M+1)
      end
  S[r]=1
   end
 return S
end

# 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,:] \cdot 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/tau
end

# 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))

# find projection to measure distances (same algorithm runned 10x longer)
xopt=x0
for t=1:10*iteration
    xproj = projection(A,b,xopt)
    xopt = (1-omega)*xopt + omega*xproj
end
print("xopt:", norm(A*xopt-b))

# 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-omega)*x + omega*xproj # project on omega
    # x = (1-bestOmega[t])*x + bestOmega[t]*xproj # project on best omega
end

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