In [1]:

```
### UCSD ECE269 Winter 2018
### Homework Set #5, Programming Assignment
### By Shouvik Ganguly

### Part 1(a): Function definition

function OMP(b, thresh, A)

    #Initialize
    residual = reshape(b, length(b), 1);
    Lambda = reshape([],1,0);
    t = 1;
    x = zeros(1, size(A)[2]);
    Anormalized = A./(sqrt.(repmat(sum(A.^2, 1), size(A)[1], 1)));
    Aused = reshape(Float64[],size(A)[1],0);

    while((t < size(A)[1])&&(vecnorm(residual) > thresh))

        newelem = indmax(abs.((Anormalized.'*residual)));
        Lambda = [Lambda newelem];
        Aused = [Aused A[:, newelem]];
        residual = (eye(size(A)[1]) - ((Aused/(Aused.'*Aused))*(Aused.')))*resid
ual;
        t = t+1;

    end

    a1 = ((((Aused).'*(Aused))\(Aused).')*reshape(b,length(b),1));

    count = 1;

    for ii in Lambda
        x[ii] = a1[count];
        count = count + 1;
    end
    x = reshape(x, length(x), 1);

    return x;

end
```

Out[1]:

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

In [2]:

```
m = 50; N = 1000; kmax = 25;
srand(1234);
pES = zeros(1, kmax); #probability of error in support
avgrelerr = zeros(1, kmax); #average relative error
numsim = 100;
thresh = 0.01;

for k = 1:kmax
    relerr = zeros(1, numsim); #relative error at each run
    supportdist = zeros(1, numsim); #support distance

    for ii = 1:numsim
        # Generate a random sparse solution x
        x = zeros(N);
        supportchosen = randperm(N)[1:k];
        xtilde = vec((rand([-1 1],1,k)).*(1+(9*rand(1,k))));
        x[supportchosen] = xtilde;

        # Random measurement of x
        A = randn(m, N);
        b = A*x;

        # OMP to find x
        xrec = OMP(b, thresh, A);

        # Compute supportdist[ii] and relerr[ii]
        supportrecovered = find(xrec);
        supportdist[ii] = (max(length(supportchosen), length(supportrecovered))
        - length(intersect(supportchosen, supportrecovered)))/max(length(support
chosen),length(supportrecovered));

        relerr[ii] = vecnorm(xrec - x)/vecnorm(x);
    end

    avgrelerr[k] = mean(relerr);
    pES[k] = mean(supportdist);

end

avgrelerr = vec(avgrelerr);
pES = vec(pES);
```

In [3]:

```
#Plot your results
using PyPlot;
ax = gca();
ax[:plot](1:25, pES,label="Probability of error");
ylabel("Probability of error");
ax[:legend](loc = "upper left", fancybox = "true");
ax2=ax[:twinx]();
ax2[:plot](1:25, avgrelerr,color="red",label="Relative error");
xlabel("sparsity");
ylabel("Relative average error");
ax2[:legend](loc="lower right", fancybox = "true")
#legend(loc = 0, fancybox = "true");
```