Description of Current Progress

NOTE: To run the program, YALL1 is needed and run “Runme_1st.m” first.

File Description:

Fmap.m : + the function handle for A(or F). Using FFT2D

Fn.m : + the function handle in YALL1 by default if the input A is a matrix

ToyCase.m:

+ aimed at constructing toy problems for proving correctness of algorithm.
+ The toy problem is of n = 100, m = 0.7*n, k = 100*100*0.001(number of sparse signals)
+ numerically proved that the Kronecker version is equivalent to the FFT2d version. Both worked well. Results very close to the exact solution. See eg results in /pics directory.
+ FFT2D is indeed faster than Kronecker version (50+s vs 0.5s on average).

BranchingProc_Demo.m:

+ code for solving transition prob using FFT2d
- somehow not work correctly. The solution is NaN or a very large X. The result is stored in Pics directory.

Data.mat:

+ data saved from R. A is **phi2d**, X is the solution **,est.accel**, given by PGD, B is **y2d,** Ind is indices.

Problem & observations:

- for toy problem, when m < 0.6*n, I observed that probability is high that there is no signal detected for both kronecker and FFT2d approach i.e. solution are simply all zeros/close to zeros. For our target problem, according to the sensing matrix A we have m = 58, n = 512. Thus the ratio is somewhat close to 0.1, which usually does not give any solution under default setting(tol = 1e-3, rho=1e-3).

- we can adjust rho, i.e. penalty for L1 as well as tolerance. But it seems that penalty does not work very well for computing X for our target problem. Because I have been tuning parameters from rho = 1 to 1e-10000, not once have I seen any reasonable solution. Xs all becomes extremely large i.e.. They are all floating around in a scale of 10^100+

Some thoughts:

I am a little bit confused at the moment. I may first try to find out if large penalty really works out since MATLAB by default supports 32 bit variable.