A. d'Aspremont
Identifying Small Mean Reverting Portfolio

Quantitative Trading Strategies

Dhanuka Rohan
Fu Xiaohang
Magnin Christophe
Sinha Prashant

Contents

- Introduction to paper
- Enhancements
- Parameter selection
- Sparse decomposition algorithms
- Greedy Search
- Semi Definite Relaxation
- Our interpretation

3

Introduction

Gist of paper and our purpose

Introduction

- Given multivariate time series, we study the problem of forming portfolios with maximum mean reversion while constraining the number of assets in these portfolios.
- We show that it can be formulated as a sparse canonical correlation analysis and study various algorithms to solve the corresponding sparse generalized eigenvalue problems, specifically, greedy search and semi definite relaxation
- After discussing penalized parameter estimation procedures, we study the sparsity versus predictability tradeoff and apply the same in two markets – FX and swaps.

Introduction

- The trading logic behind this paper is that if we confidently predict mean reversion of a basket of assets, we can then use that property to trade those assets as we have knowledge of their long term trends.
- The problem in this paper also deals with number of assets. While increasing number of assets in the basket increases the mean reversion, it may reduce financial viability of trading due to complexity and cost. Thus, we want to find a solution amongst as few assets as possible.
- We thus create a asset number vs. mean reversion trade off analysis to decide optimal asset number for a given set.

Introduction

- Once we have the assets, we can apply it to a real time trading environment and see if we can make a profit.
- We go about this task using a number of complex mathematical tools, especially in Matlab. The exact implementation is discussed next and all the code is provided so that our implementation is immediately usable.

Introduction

- Deviation from the average price are expected to revert to the average
  - Sell (buy) when asset higher (lower) than average
- Classis indicator of predictability in financial markets
- Usual methods (cointegration, etc) to get mean reverting portfolios give dense portfolios
  - High transaction costs
  - Issues to interpret the resulting portfolios
- Goal of d'Aspremont: Finding a way to get sparse mean reverting portfolios
  - Given a (high) number of assets, we build a portfolio with *a few* of them by maximizing predictability

Enhancements

What we've added beyond the paper

Enhancements

- The biggest issue for us was that there was low real world application to this paper, as he proves that this technique does not yield profits once transaction cost is allowed for.
- We wanted to find a way to make our efforts have real world dividends. We decided to do this by making use of Excel.
- Our purpose is to provide a trader with a black box solution where in he simply clicks a button and gets information on the mean reversion property of the assets chosen.
- He can also change the covariance matrix as per his views.

Enhancements

- Working in a trading environment, we knew that ease of use and stability was desirable as much as final results.
- We propose a daily autosys job to run the parameter selection matlab code which uses bloomberg data and stores results in a shared drive.
- The trader in the day simply starts excel which can import data from the share drive. Then, the trader presses calculate and gets a complete detailed picture of the mean reversion properties of different cardinality asset baskets.
- He can now use this information to support his other mean reversion trading strategies.

11

Parameter Selection

How to get the parameters required. Uses Matlab.

Covariance Selection

- Robust and sparse estimation of the covariance matrix developed by Dempster (1972)
- Maximum-likelihood penalized by the cardinality of X

$$\max_x \log\det X - \mathrm{Tr}(\sum X) - \rho\,\mathrm{Card}(X)$$

  - in the variable $X \in S_n$, where $\boldsymbol{\Sigma}$ is the sample covariance matrix, *Card(X)* is the number of nonzero coefficients in X and $\boldsymbol{\rho}$>0 controls the trade-off between log-likelihood and sparsity
- Hard to solve numerically. D'Aspremont, Banerjee & El Ghaoui (2006) replaces the penalty by the ℓ_1 norm

$$\max_x \log\det X - \mathrm{Tr}(\sum X) - \rho \sum_{i,j=1}^{n} |X_{ij}|$$

Covariance Selection – Dual Problem Block-coordinate descent gradient method

- The dual problem is given by

$$\min -\log\det(\Sigma + U) - n$$

$$\text{subject to} \; |U_{ij}| \leq \rho, \quad i, j = 1, \ldots, n$$

  - in the variable $U \in S_n$

- We can decompose the matrices in blocks as follows

$$U = \begin{pmatrix} V & u \\ u^T & w \end{pmatrix} \quad \text{and} \quad \Sigma = \begin{pmatrix} A & b \\ b^T & c \end{pmatrix}$$

  - where V is fixed, $A \in S_{(n-1)}$, $u,b \in R^{(n-1)}$ and $w,c \in R$
  - u represents the variables, i.e. the rows and columns we are updating at each iteration.

Covariance Selection – Dual Problem Block-coordinate descent gradient method

- The dual problem in blocks becomes

$$\min -\log\det(A+V) - \log[(w+c) - (b+u)^T (A+V)^{-1}(b+u)] - n$$
$$\text{subject to} \; |w| \le \rho, \; |u_{ij}| \le \rho, \quad i, j = 1, \ldots, n$$

- At each iteration, the main step is then a box constrained quadratic program of the form

$$\min (b+u)^T (A+V)^{-1}(b+u)$$
$$\text{subject to} \; |u_{ij}| \le \rho, \quad i, j = 1, \ldots, n$$

  - can be solved using SeDuMi by Sturm (1999)

Block-coordinate descent gradient method Algorithm

- Pick the row and colum to update;
- Compute $(A + V)^{-1;}$
- Update the row and column previously picked with the solution of the box constrained QP
- After each step, check the folowing convergence condition

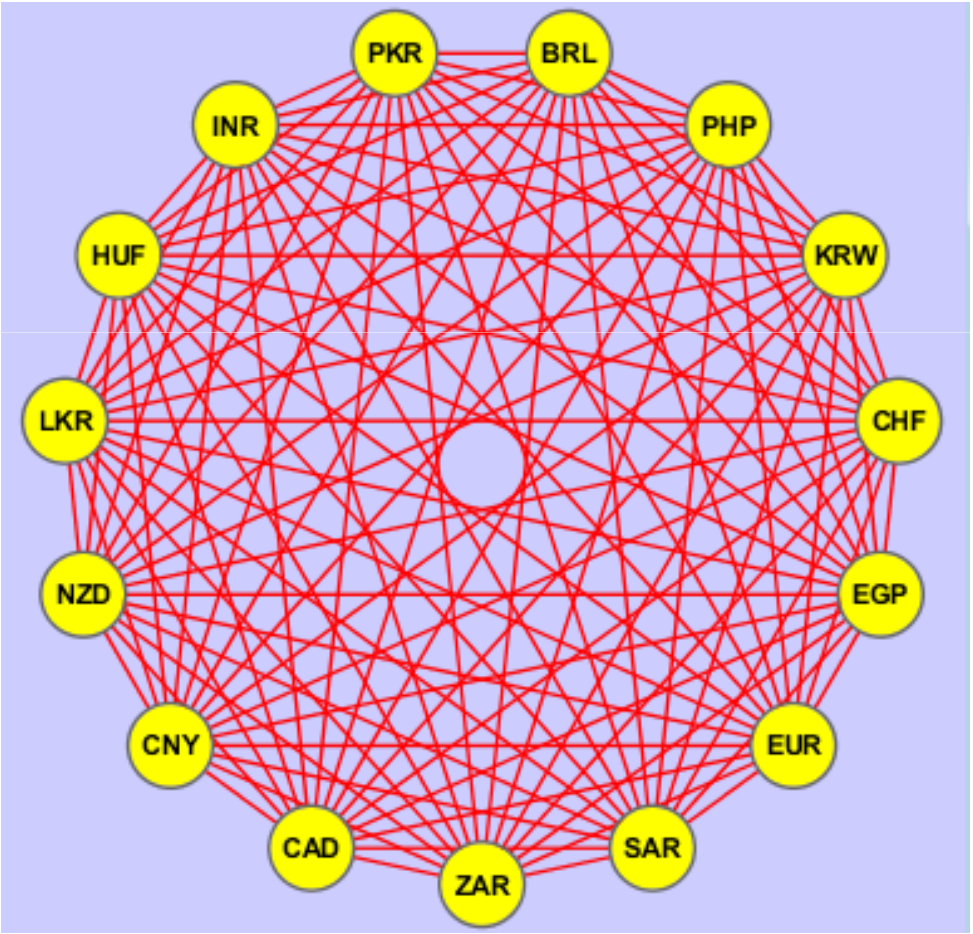
$$\mathrm{Tr}(\sum X) - n + \rho \sum_{i,j=1}^{n} \left| X_{ij} \right| \leq \varepsilon$$

  - Matlab code available on d'Aspremont's website

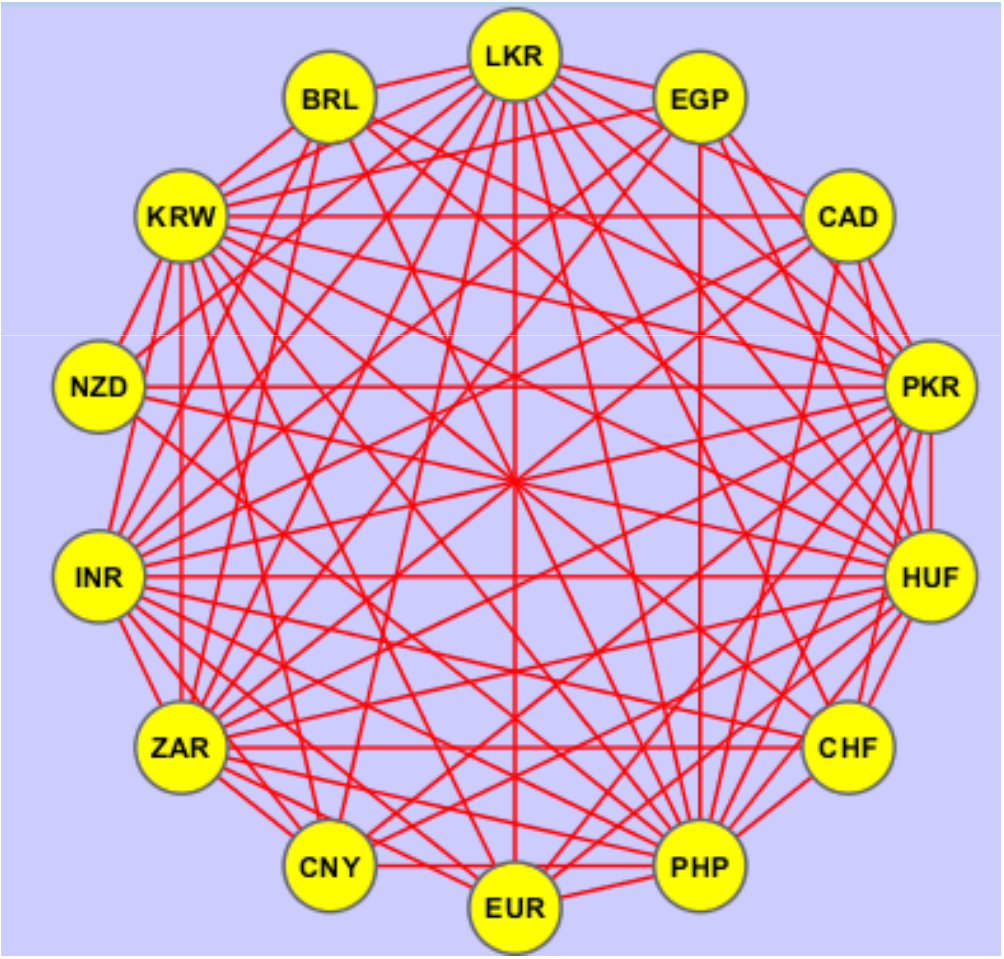
Covariance Selection Example

- Each component equal to 0 in the inverse covariance matrix represents two variables that are conditionally independent
  - Financial meaning of the inverse covariance matrix: idiosyncratic components of asset prices dynamics
- Example
  - 15 currencies vs. USD
  - daily data from Jan 2008 to Dec 2009
  - graphs done using Cytoscape

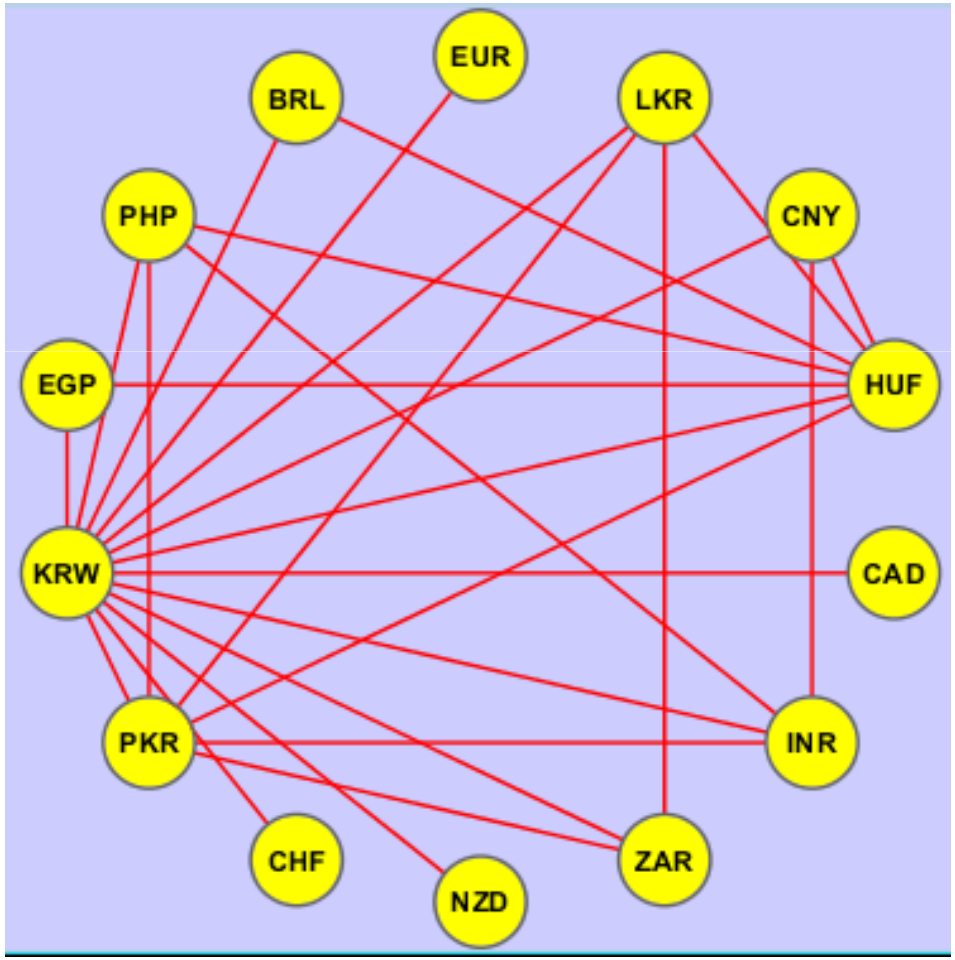
Covariance Selection
Initial inverse covariance matrix



Covariance Selection
rho = 0.05



Covariance Selection
rho = 0.5



A matrix

- Asset price is an autoregressive process. The second parameter to estimate is the A matrix

$$S_t = S_{t-1}A + Z_t$$

  - where S_{t-1} is the lagged portfolio process, $A \in R^{nxn}$ and Z_t is a vector of Gaussian noise independent of S_{t-1} with zero mean and covariance $\mathbf{\Sigma} \in S^n$
- Intuitive solution: ordinary least square

$$\widehat{A} = (S_{t-1}^T S_{t-1})^{-1} S_{t-1}^T S_t$$

  - OLS estimate is used in the Box & Tiao procedure.

Sparse A matrix
Exogeneous dependence model

- It is possible to get a sparse version of the A matrix. Similarly to the covariance selection, we use a penalized estimator

$$a_i = \arg\min_x \left\| S_{it} - S_{t-1x} \right\|^2 + \gamma \left| x \right|_1$$

  - in the variable x ϵR^n and where γ > 0 controls sparsity
  - a_i gives the column i of the matrix A
- This is known as the LASSO and can be solved using different algorithms.
  - We used a method developed by Fu (1998) and called the shooting method
    - We modified a Matlab package available at http://www.cs.ubc.ca/~schmidtm/Software/lasso.html

Sparse Decomposition Algorithms

Discussing the concepts

Sparse Decomposition Algorithms

- The nub of this problem is to solve for a generalized eigenvalue problem which can be represented as

$$\det(\lambda B - A) = 0$$

in the variable $\lambda \in R$, where $A,B \in Sn$. This is usually solved using a QZ decomposition. The largest solution of this problem can be written in variation form as:

$$\lambda^{\max} = \max_{x \in \mathbb{R}^n} \frac{x^T A x}{x^T B x}.$$

Sparse Decomposition Algorithms

- It is this lambda that we seek to maximize. We need to do this within certain constraints we introduce for financial reasons such as transaction cost. The main constraint we introduce is one of cardinality and finally, we solve the following

$$\begin{array}{ll} \text{maximize} & x^T Ax / x^T Bx \\ \text{subject to} & \mathbf{Card}(x) \leq k \\ & \|x\| = 1, \end{array}$$

Here, is a constant greater than 0 while card(x) represents the number of non zero assets. We are basically constraining the number of assets to trade for that max lambda.

Background

- Problem

  estimating sparse mean reverting portfolios:

  assume portfolios x_i follow an Ornstein-Uhlenbeck process: $dP_t = \lambda(\overline{P} - P_t)dt + \sigma dZ_t$

  with $P_t = \sum_{i=1}^{n} x_i S_{ti}$

  S_{ti} is the value of asset x_i at time t; Portfolio P_t is formed by these assets with weights S_i

- Objective

  maximize mean reversion coefficient

  by adjusting portfolio weights x_i

  Under constraint : $\|x\| = 1$

  cardinality of x below k.

Sparse Decomposition Algorithms

- As a result of canonical decomposition, we get:

$$\lambda^{\max}(A,B) = \max_{x \in R^n} \frac{x^T A x}{x^T B x}$$

- Thus, we get a model considering of the cardinality constraint:

$$\text{maximize} \quad x^T A x / x^T B x$$
$$\text{subject to} \quad Card(x) \le k$$
$$\|x\| = 1$$

- Next, we discuss two techniques to get approximate solutions:

1. ***Greedy Search***
2. ***Semidefinite Relaxation***

Canonical Decomposition -- 1

- Assume asset prices serie follows a stationary vector autoregressive process:

$$S_t = S_{t-1}A + Z_t$$

measuring predictability by variance ratio:

$$v = \frac{\sigma_{t-1}^2}{\sigma_t^2}$$

- As for the portfolio, we measure predictability by

$$v(x) = \frac{x^T A^T \Gamma A x}{x^T \Gamma x}$$

Canonical Decomposition -- 2

- As for the portfolio:

minimizing predictability

equivalent to

finding minimum generalized eigenvalue by

$$\det(\lambda\Gamma - A^T\Gamma A) = 0$$

- Portfolio with minimum predictability is:

$$x = \Gamma^{1/2} z$$

z is the eigenvector corresponding to the smallest eigenvalue of

$\Gamma^{-1/2} A^T \Gamma A \Gamma^{-1/2}$

A is estimated by: $\hat{A} = (S_{t-1}^T S_{t-1})^{-1} S_{t-1}^T S_t$

Greedy Search

Implemented completely as a black box solution in Excel

Greedy Search

- The concept of greedy search is exceedingly simple and leads to a solution which might not be the most optimal but is highly cost efficient. We find the first optimal point, then the optimal point closest to it and so on.
- In this particular execution, we first find the single asset giving the highest mean reversion. We then fix this asset and search for the asset amongst the others such that the pair gives the highest mean reversion. In this way, we keep adding assets till k is reached.

Greedy Search

- We start with a definition

$$I_k = \{i \in [1, n] : \ x_i \neq 0\}$$

- We now want to build recursive solutions with regards to k. We first find the initial starting point with the equation,

$$I_1 = \underset{i \in [1,n]}{\operatorname{argmax}} A_{ii}/B_{ii}.$$

- We now start adding weights to assets one by one for maximum predictability as explained earlier. The equation solved is

$$\max_{\{x \in \mathbf{R}^n:\ \operatorname{supp}(x) = I_k \cup \{i\}\}} \frac{x^T A x}{x^T B x}$$

Greedy Search

- The cost of the solution for this implementation is O(n^4) for all k which is far more efficient than many of the deep search algorithms implemented for the problem.
- There are however some slight stability issues with this algorithm.
- Also, a major issue encountered during implementation as well is that the optimal solution for the equation solved might not have an increasing support set as is assumed by the algorithm which implements $I_k \subset I_{k+1}$.
- Thus our final answer while efficient is rarely the most optimal and so we look at other algorithms for verification.

Implementation of Greedy Search

- In our implementation of greedy search, we wanted to find a more trader or front office friendly solution, especially as this was the simpler of the two algorithms.
- We thus decided to use Excel and VBA coding for our final solution. Our concept is that we can calculate the matrices using Matlab as a singular package and store them in a library.
- Then, at any point, the trader could import the data, modify as required and by simply clicking one macro, he would get the optimal weights with associated mean reversions, allowing him to take a call.

Implementation of Greedy Search

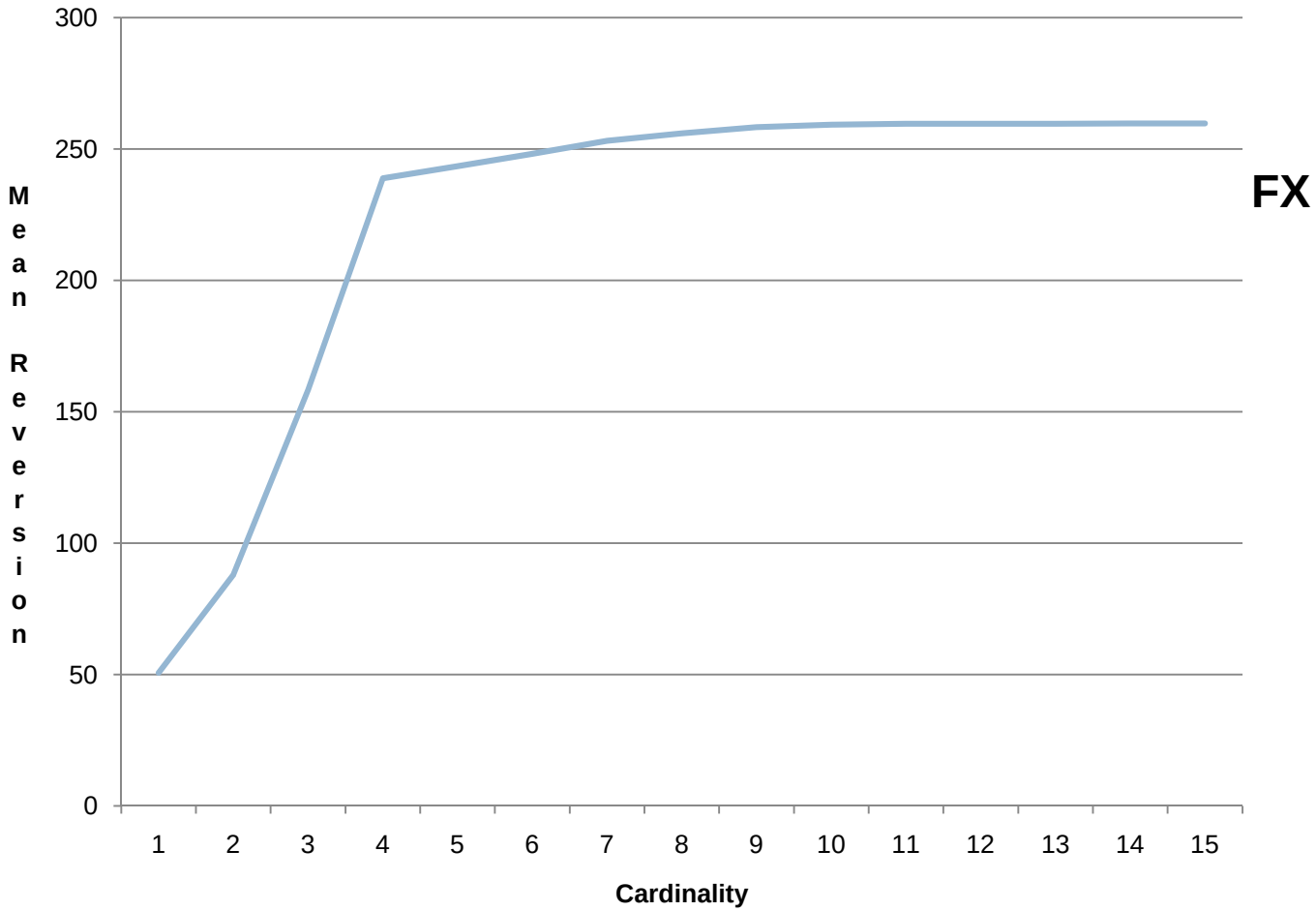
- Using this method, this paper becomes more useful as it provides a weighted and fundamental research into asset mean reversion and allows the traders to make real time calls.
- The reason we did this was using only d'Aspremont's method, we found that there is no profit being generated after including transaction cost. We then searched for a way to make this ore useful for the final user and came up with this.
- For excel execution, we use the Matrix functions inbuilt in excel and extensively use the Solver with constraints as specified to generate the final values. It fundamentally follows the greedy search algorithm in all aspects.

Result of Greedy Search

- We ran two greedy searches. One was on FX and one on Swap rates over 50 time windows. The result of the FX was found to be-

Cardinality	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Asset 1	0	0	0	0	0	0	0.129296	0.135693	0.16219	0.162149	0.161749	0.161798	0.163131	0.163441	0.163441
Asset 2	1	0.999238	0.503557	0.239637	0.256918	0.307738	0.339188	0.336403	0.312109	0.310968	0.310583	0.311346	0.308943	0.309391	0.309391
Asset 3	0	0	0	0	0	0	0	0	0	0	0	-0.00125	-0.00141	-0.00145	-0.00145
Asset 4	0	0	0	0	-0.05196	-0.07825	-0.0938	-0.09098	-0.08456	-0.08555	-0.0845	-0.08846	-0.08681	-0.08728	-0.08728
Asset 5	0	0	0	0	0	0	0	0	0	0	0	0	-4.7E-05	-3.8E-05	-3.8E-05
Asset 6	0	0	0	0	0	0	0	0	0	0.078145	0.079008	0.079097	0.080591	0.079788	0.079788
Asset 7	0	0	0	0	0	0	0	0	0	0	0.041003	0.041316	0.041039	0.040987	0.040987
Asset 8	0	0	0	0	0	0.036282	0.041344	0.039085	0.040186	0.036361	0.035715	0.037818	0.037867	0.03831	0.03831
Asset 9	0	0	-0.86094	0.970465	0.964887	0.946306	0.924632	0.916597	0.921356	0.918434	0.917843	0.917257	0.91802	0.917737	0.917737
Asset 10	0	0	0	0	0	0	0	0.126533	0.12506	0.127232	0.126797	0.126124	0.125667	0.125765	0.125765
Asset 11	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.0092	-0.0092
Asset 12	0	0	0	-0.00634	-0.00477	-0.00581	-0.00696	-0.00754	-0.00751	-0.00748	-0.00756	-0.00753	-0.00744	-0.00745	-0.00745
Asset 13	0	0.039021	0.072193	-0.02705	-0.01623	-0.04825	-0.05227	-0.04896	-0.05212	-0.04805	-0.04775	-0.04641	-0.04481	-0.0453	-0.0453
Asset 14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Asset 15	0	0	0	0	0	0	0	0	-0.01441	-0.01366	-0.01365	-0.0137	-0.01328	-0.01344	-0.01344
Lambda	50.60261	87.93564	158.4448	238.8242	243.4302	248.1006	253.1187	255.8817	258.2391	259.2139	259.5234	259.592	259.6226	259.6345	259.6345

Results of Greedy Search

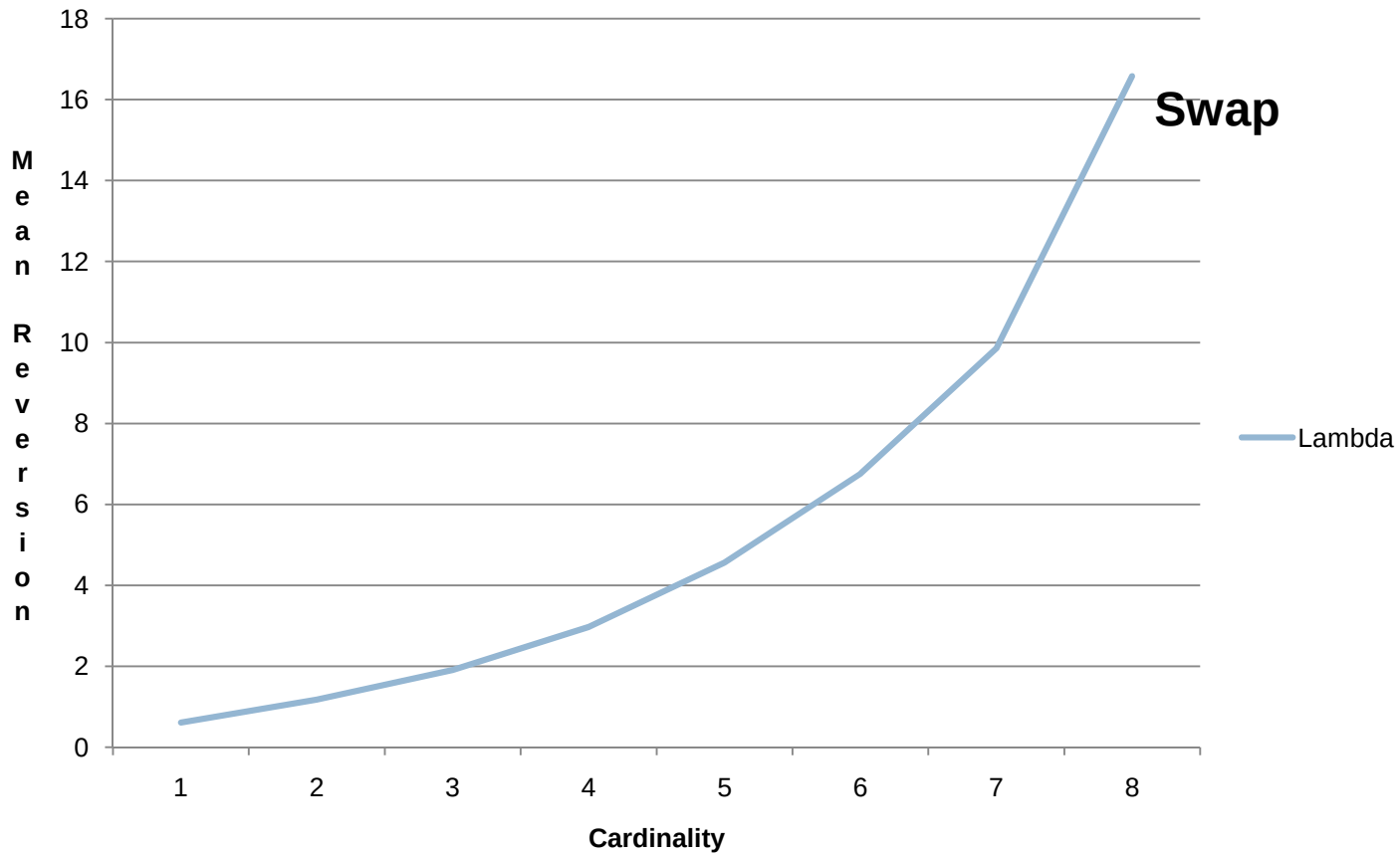


Results of Greedy Search

- Similarly for FX, we found the average of the mean reversions given cardinalities over all time windows.

Cardinality	1	2	3	4	5	6	7	8
Lambda	0.6158	1.182	1.913	2.9735	4.562	6.7554	9.854	16.58

Results of Greedy Search



Semi Definite Relaxation

Complete implementation in Matlab

Semidefinite Relaxation -- 1

- Formulate a semidefinite relaxation for sparse generalized eigenvale problems:

$$\text{maximize } x^T A\, x / x^T B\, x$$
$$\text{subject to } \mathrm{Rank}(X) = 1$$
$$\|x\| = 1$$
$$\text{with } x \in R^n$$

- Equivalent problem (let $X = xx^T \in S^n$):

$$\text{maximize } \mathrm{Tr}(AX)/\mathrm{Tr}(BX)$$
$$\text{subject to } \mathrm{Card}(X) \le k^2$$
$$\mathrm{Tr}(X) = 1$$
$$\mathrm{Rank}(X) = 1$$
$$X \succeq 0$$

Semidefinite Relaxation -- 2

- Furthermore, we can relax the cardinality constraint by a weaker convex constraint, and

get the relaxed model:

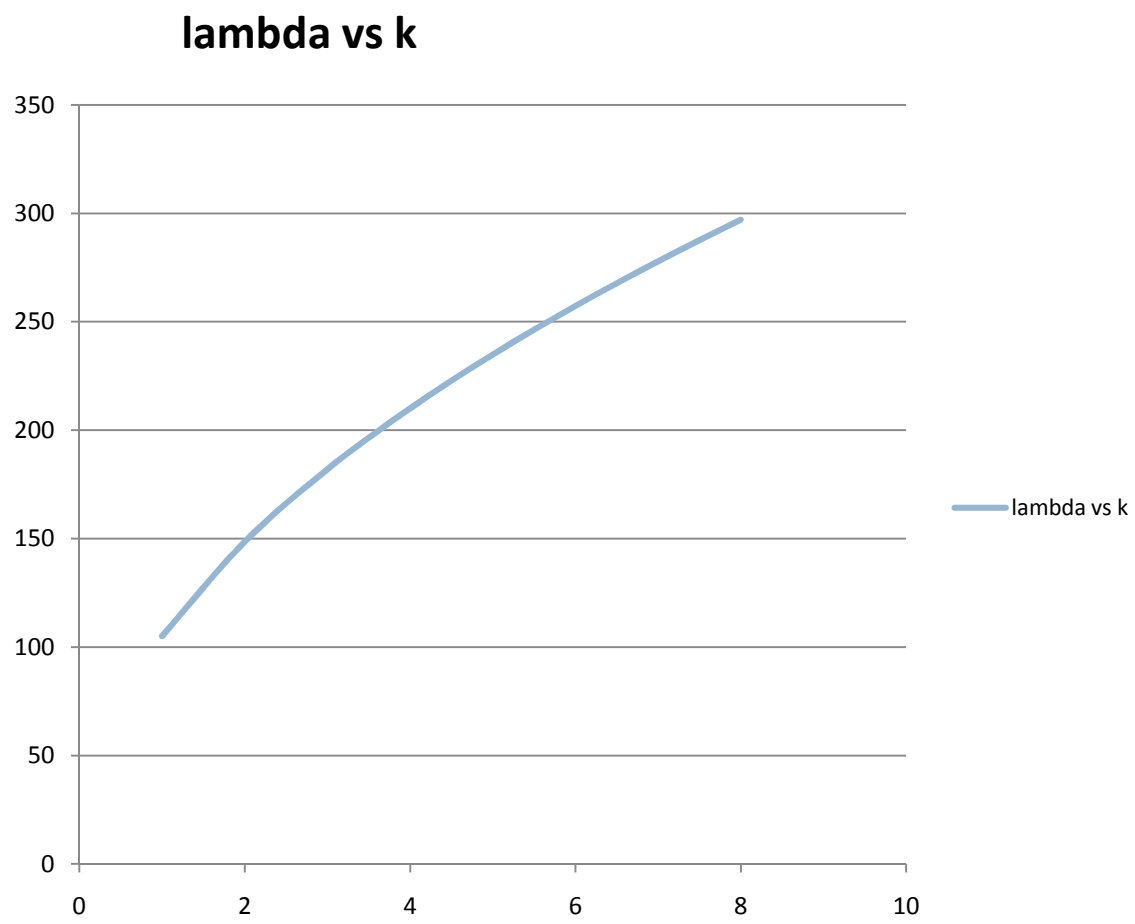
$$\begin{aligned} &\text{maximize } \mathrm{Tr}(AX)/\mathrm{Tr}(BX) \\ &\text{subject to } 1^{\mathrm{T}}|X|1 \le k \\ &\qquad \mathrm{Tr}(X) = 1 \\ &\qquad X \succeq 0 \quad \text{with } X \in S_n \end{aligned}$$

- By changing variable: $Y = \frac{X}{Tr(BX)}$ $z = \frac{1}{Tr(BX)}$

$$\begin{aligned} &\text{maximize } \mathrm{Tr}(AY) \\ &\text{subject to } 1^{\mathrm{T}}|Y|1 - kz \le 0 \\ &\qquad \mathrm{Tr}(BY) = 1 \\ &\qquad Y \succeq 0 \quad \text{with } Y \in S_n \text{ and } z \in R_+ \end{aligned}$$

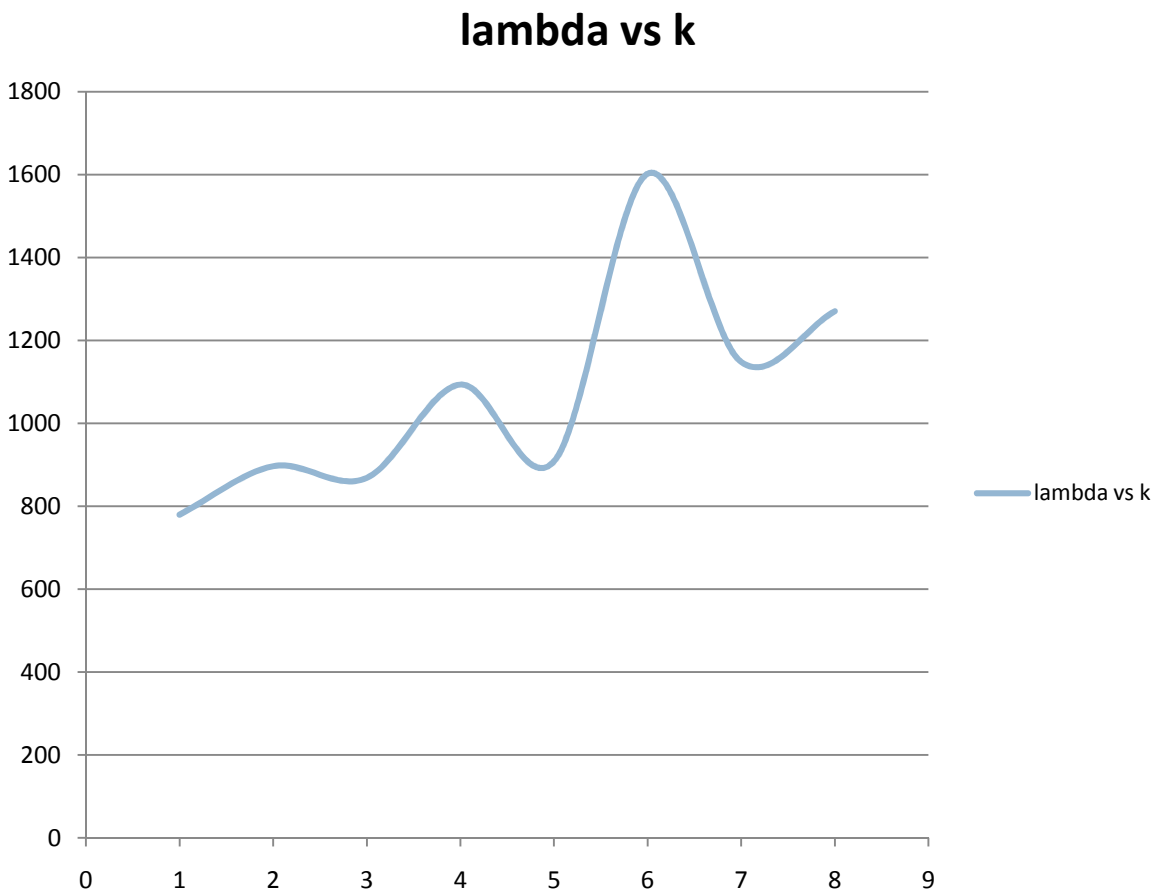
Semidefinite Relaxation
Results for Swap rate Data

k	lambda
1	105.0539
2	148.5687
3	181.9588
4	210.1079
5	234.9077
6	257.3285
7	277.9466
8	297.1374



Semidefinite Relaxation Results for FX rate Data

k	lambda
1	779.802
2	896.8612
3	868.4883
4	1093.477
5	908.2806
6	1601.532
7	1148.023
8	1270.215



Intepretation

Our interpretation of the entire project

Interpretation

- We were impressed with the mathematical rigor of the paper and the strong techniques used to implement it. However, we were slightly unhappy with the end result.
- As in the paper, the final results are not strong enough to be used as a all in trading technique. What we instead propose is to simplify the execution of this process and use it as a fundamental analysis tool to help support the trader's decision making
- Greedy search was found to be almost as good at interpretation as the much more costlier semi definite relaxation

Interpretation

- We would propose we use this project's implementation to work on structured products rather than vanilla ones. We could create worst of equity baskets or other structured products what rely on dependence between the component assets.
- As a pure analysis tool, the d'Aspremont technique is pretty robust and better than a couple of competing off the shelf trend analysis tools we found online.
- We also feel we had a good take away in terms of Matlab package knowledge and core algorithm creation.

References

- Banerjee O., El Ghaoui L., d'Aspremont A., *Model Selection through Sparse Maximum Likelihood Estimation*, Journal of Machine Learning Research, pp. 485-516, March 2008
- d'Aspremont A., *Identifying Small Mean Reverting Portfolios*, Quantitative Finance, pp. 351-364, March 2011
- Fu W. J., *Penalized Regressions: The bridge versus the Lasso*, Journal of Computational and Graphical Statistics, pp. 397-416, September 1998