Algorithmic Finance 3 (2014) 173–188
DOI 10.3233/AF-140038
IOS Press

Fast recursive portfolio optimization

Laurence Irlicht*
Investment Director, Indexed Equities, IFM Investors
Adjunct Professional Fellow, Dept. of Banking & Finance, Monash University, Melbourne, Australia

Abstract. Institutional equity portfolios are typically constructed via taking expected stock returns and then applying the computationally expensive processes of covariance matrix estimation and mean-variance optimization. Unfortunately, these computational costs make it prohibitive to comprehensively backtest and tune higher frequency strategies over long histories. In this paper, we introduce a recursive algorithm which significantly lowers the computational cost of calculating the covariance matrix and its inverse as well as an iterative heuristic which provides a very fast approximation to mean-variance optimization. Together, these techniques cut backtesting time to a fraction of that of standard techniques. Where possible, the additional step of caching pre-calculated covariance matrices, can result in overall backtesting speeds up to orders of magnitude faster than the standard methods. We demonstrate the efficacy of our approach by selecting a prediction strategy in a fraction of the time taken by standard methods.

Keywords: Portfolio optimization, algorithmic finance, covariance estimation, quadratic optimization, computational finance, mathematical programming, Backtesting

1. Introduction

Many institutional equity strategies, as well as much of the published empirical finance research have been analyzed using annual, monthly or much less often weekly data. See for instance Fama and French (2012), Chan et al. (1996), Liu et al. (2011) and Rachev et al. (2007) for momentum strategies, Poterba and Summers (1988) for reversion, Fama and French (1992) and Haugen and Baker (2010) for style analysis, as well as Scherer (2011), Haugen and Baker (1991), Arnott et al. (2005) and Amenc et al. (2010) for 'Smart Beta' approaches.

The proliferation of higher frequency data, increasing computer storage and power, and increasingly crowded quantitative trades, Brown (2010), all raise the question as to how such strategies may be enhanced via the analysis of daily or even higher frequency data.

One bottleneck which hinders such analysis is the increased computational effort required over such frequencies. For instance, a daily analysis will have twenty times the data points of a monthly analysis over the same period.

Common practice for institutional equity portfolio design employs constrained mean-variance optimization. Equality constrained mean-variance problems can be efficiently solved analytically, however the practical necessity for inequality constraints (such as active weight limits or non-shorting rules) adds sufficient complexity to require numerical solution. This is computationally expensive and hence much slower than the analytic approach.

The time to calculate one portfolio is less than one second, and therefore does not matter for the purpose of live investment for all but high frequency strategies. However, when we seek to backtest strategies over extensive time periods with large numbers of parameters to be set - it does matter. Critically.

The more efficiently we can backtest, the more scope we have for sensitivity and robustness analysis and parameter tuning. To be clear, we are not advocating

*Corresponding author: Laurence Irlicht, IFM Investors, Level 29, 2 Lonsdale Street, Melbourne 3000, Australia. Tel.: +613 8672 5462; E-mail: Laurence.Irlicht@ifminvestors.com.

mindlessly searching for strategies that happened to work by chance, but systematically and robustly analysing strategies within training data sets, and then verifying them over separate evaluation data sets.

For instance, consider a back-test using ten years of daily data on a portfolio which holds 150 stocks. This would involve the calculation of around 2,400 150 by 150 covariance matrices, and also 2,400 constrained optimizations. We tested this on a Windows 7 desktop, powered by an Intel i7 CPU @ 2.8 GHz running Python 2.7.2 and Numpy 1.6.2. It took over 10 minutes. Whilst this speed could be improved via a faster processor or a faster language such as C, the techniques we introduce here will speed up the computation regardless of the processor or language used.

For time critical applications, we would suggest using these techniques in conjunction with the fastest available processor and language, and where possible distributing the calculations among many processors.

If our strategy had many parameters (for instance parametrised methods of estimating each stock's earnings quality, momentum, value, liquidity and expected growth) and we needed to see the impact of changing such parameters together, then Bellman's curse of dimensionality comes into play and it is clear that anything but highly efficient techniques quickly become bogged down and unusable - infeasibly slow.

The problem is that the standard portfolio optimization methods which involve both a full calculation of the covariance matrix as well as an inequality constrained quadratic optimization at each iteration are computationally expensive.

One way often used to avoid this issue is to replace the mean-variance optimization with simpler (faster) ad-hoc weighting schemes. This is most commonly applied to higher frequency trading. Examples of such include the equal weighting of quantiles or pairs and other risk agnostic weighting schemes. See for instance Do and Faff (2010) or Schulmeister (2009).

Whilst they may add value, such schemes ignore the risk information inherent in the covariances of the securities involved and as such may lead to sub-optimal (higher risk than necessary) portfolios.

Alternatively, mean-variance analysis simultaneously takes account of both risk and return, and can provide the ex-ante lowest risk portfolio for a given level of return. However, as shown above, such analysis (using standard methods) is prohibitively computationally expensive for backtests over large numbers of periods for large numbers of stocks. It just takes too long.

What we seek is a method by which we can explore mean-variance strategies at higher frequencies, and do so in a timely manner.

We begin by recalling the nature of equity portfolio optimization.

Key is the fact that the signal to noise ratio in equity return prediction is very low, Kritzman (1998). This is also evidenced by the typically low information ratios managers can generate, Goodwin (1998). Ex-ante alphas (predicted returns) and covariance matrix estimates are inherently noisy, as are 'optimal' portfolios calculated from them.

Furthermore, even if we had accurate alphas and covariance matrices, a further source of noise is the fact that history is only an approximate proxy for the future. They may be accurate for the past, but due to systemic changes, inaccuracy created by moving from a backtest to future (real investment) is likely to be considerable.

These unavoidable inaccuracies mean that it is false confidence to believe that our backtests are extremely accurate. However, acknowledging that such tests are inaccurate, a premise of quantitative investing is that there is still sufficient accuracy to differentiate a bad strategy from a good one.

With this in mind, we advocate the use of fast but approximate optimization procedures. They notably speed up the backtest process, and when applied correctly, the noise they add is immaterial in light of the other sources of noise in the system. Indeed, once interesting strategies have been detected they can be re-analysed using exact methods. This is reminiscent of types of search algorithms such as adaptive step size or adaptive mesh models which use large steps to map out the territory, and smaller steps where more accuracy is necessary. See for instance Figlewski and Gao (1999) or Viveros-Jimnez et al. (2013).

Thus we propose the use of three different optimization methods. An extremely fast, but more approximate method for an initial search (Analytic), a fast and more accurate method once the general region of parameter space has been selected (Iterated Analytic). Finally, exact, but slow, methods can be used to double check the proposed strategies (Full Calculation). These optimization methods are outlined below:

- **Optimization method 1: Analytic**
  The fastest procedure is to simply ignore any inequality constraints and use an analytic solution for the equality constrained portfolio. This

provides a very fast and totally accurate solution to the equality constrained case, but is inaccurate for the inequality constrained case (for instance requiring all stock weights to be greater than zero). However, it can still provide a good measure of the efficacy of a strategy in that the measurement error in information ratio caused by the analytic approximation will tend to be much less than the difference in information ratios between a good and a bad strategy. This is explored further in Section 4 which details some empirical analysis.

- **Optimization method 2: Iterated analytic**
  This mid-range method still offers significant speed improvements over the full optimization case, but with enhanced accuracy (and adherence to inequality constraints). It involves the iterative use of analytic solutions, but where the optimization problem is modified to cause the satisfaction of the inequality constraints. This method takes only a fraction of the time of using an exact optimization and provides investable portfolios.
- **Optimization method 3: Full calculation**
  Once the parameter region of interest has been identified via the Analytic or Iterated Analytic methods above, a full optimization is undertaken to ensure that the results hold in the 'exact' case.

The optimization methods above require either the covariance matrix (full calculation) or the inverse of the covariance matrix (analytic and iterated analytic methods). These calculations themselves can be time consuming, especially for large portfolios. However, this delay can be avoided.

Engineering applications routinely estimate parameters via computationally efficient recursive methods, for instance Young (2011) or Anderson and Moore (1979). This leads to great efficiency gains when compared to a complete re-estimation every time period. The problem is that such techniques cannot be used for our purpose without modification.

Engineering applications typically have a fixed set of variables (for instance state variables of a machine) and thus the recursive techniques employed are restricted to such situations. On the other hand, portfolios of stocks have a changing set of variables since each 'variable' represents a stock, and stocks are continuously joining or leaving the index.

To account for this, we extend these recursive methods to allow for changing variables. This gives us an efficient method by which to calculate the covariance matrix, and also to directly calculate the inverse of the covariance matrix (without a computationally expensive matrix inversion) in the presence of changing variables and/or exponential smoothing.

The (inverse) covariance matrix calculated using our methods is identical to that calculated in full using the whole data set, but the calculation may be an order of magnitude faster.

The relative speed will be affected by issues such as the frequency of variable (stock) changes, the number of periods in the estimation window, and the number of variables (stocks). Generally, the fewer the number of stock changes and the larger the window length and number of stocks, the greater the speed-up.

Additionally, we advocate the use of cached precalculated covariance matrices (or inverse covariance matrices) where such is feasible. This again provides a significant speed up, even over the fast recursive methods we propose.

Similar to the optimization techniques, the faster covariance techniques are advocated for a broad search over parameter space, with the slower (exact) techniques used in interesting regions where more accuracy is required. The difference is that the caching of precalculated matrices can provide extremely quick, and perfectly accurate (or at least as accurate as possible given the vagaries of the data) matrices.

Thus, the covariance (inverse covariance) estimation procedures we propose are as follows:

- **Covariance method 1: Fast recursive estimation**
  The fastest (non cached) method involves a recursion on the (inverse) covariance matrix. In the case of exponential smoothing with a fixed window length, we utilise an approximation which speeds up the calculation at the cost of non-exact results.
- **Covariance method 2: Full calculation**
  The slow but exact method involves the calculation of the covariance matrix at each time step, followed where necessary by the inversion of that matrix (since the inverse of the covariance matrix is an input to the analytical optimization formula).
- **Covariance method 3: Cached estimation**
  Here the (inverse) covariance matrices are calculated and stored during the first iteration of the backtest. The stored matrices are then used for

subsequent iterations, providing the accuracy of the full calculation along with even faster speed than the recursive estimation process.

When coupled with analytical or iterated analytical approaches to mean variance optimization, these techniques offer up to orders of magnitude faster analysis of portfolios when compared to standard methods which use a new calculation of the covariance at each update, and a full (constrained) optimization.

The remainder of the paper is set out as follows. Section 2 details the very fast analytical formula for equality constrained mean-variance optimization and the fast iteration for inequality and equality constrained mean-variance optimization. Section 3 provides online (recursive) methods for the estimation of the covariance matrix and inverse of the covariance matrix. Empirical studies which demonstrate the speed and accuracy of our approach on real data are included in Section 4. Conclusions follow in Section 5.

2. Fast optimization

To introduce notation, define $\mathbf{r}$ as a column vector of the expected returns of a set of stocks. Return is defined as price at time $t+1$ divided by price at time t, -1. Σ is the covariance of returns for several stocks, and $\sum$ represents the mathematical summation operator. $\mathbf{w}$ is a column vector of portfolio weights, and λ is the risk aversion parameter - a higher λ represents a higher relative importance of risk relative to return. $\mathbf{1}$ is a column vector of ones, and $\mathbf{0}$ is a column vector of zeros, both the same dimension as $\mathbf{r}$.

We will assume that the portfolio's investment objective can be expressed in the form of maximising the portfolio's expected return minus λ times its expected variance, as $\max\{w'r - \frac{\lambda}{2} w'\Sigma w\}$.

The portfolio will also be subject to various constraints. For instance, typical constraints for an institutional portfolio may involve no short sales ($\mathbf{w_i} > 0 \; \forall \; i$), maximum stock or sector absolute weights or deviations from a benchmark, turnover limits or a sum-to constraint on all weights to ensure appropriate investability. For instance full investment with a sum to one constraint, or long-short investment with a sum to zero constraint.

2.1. Analytic approach

If we specialise to the case of equality constraints only, then the solution of the optimization problem can be expressed analytically. This is a well known result, and forms the beginning of our approach. For completeness we express it in the form of Lemma 2.1.

Lemma 2.1. *Consider the portfolio optimization problem of maximising the expected return minus λ times the variance subject to equality constraints. Then we have:*

$$\begin{array}{l}\arg\max_{\mathbf{w}} \left\{\mathbf{w}'\mathbf{r} - \frac{\lambda}{2}\mathbf{w}'\Sigma\mathbf{w}\right\} \\ \text{subject to} A\mathbf{w} = \mathbf{b}\end{array} = \frac{1}{\lambda}\Sigma^{-1}\left[I - A'\left(A\Sigma^{-1}A'\right)^{-1}A\Sigma^{-1}\right]\mathbf{r} + \Sigma^{-1}A'\left(A\Sigma^{-1}A'\right)^{-1}\mathbf{b} \quad (1)$$

Furthermore, in the case where we have a sum-to constraint on the weights, and no other constraint, then Equation (1) specialises to:

$$\begin{array}{l}\arg\max_{\mathbf{w}} \left\{\mathbf{w}'\mathbf{r} - \frac{\lambda}{2}\mathbf{w}'\Sigma\mathbf{w}\right\} \\ \text{subject to} \mathbf{1}'\mathbf{w} = b\end{array} = \frac{1}{\lambda}\left[\Sigma^{-1}\mathbf{r} - \left(\frac{\mathbf{1}'\Sigma^{-1}\mathbf{r} - \mathbf{b}\lambda}{\mathbf{1}'\Sigma^{-1}\mathbf{1}}\right)\Sigma^{-1}\mathbf{1}\right] \quad (2)$$

The proof of Lemma 2.1. can be seen in the appendix.

Whilst limited in applicability, the analytic approach has the great benefit of speed. It only requires a few matrix additions and multiplications and at most two inverses - those of Σ, and of $A\Sigma^{-1}A'$. These can be done very efficiently:

- The only matrix inverse required for the single constraint case of Equation (2) is Σ^{-1}. In the next section we show that this inverse can be updated, both for new data and also for dynamically changing stock inclusions, very efficiently in a recursive manner.
- Equation (1) also requires the inverse of $A\Sigma^{-1}A'$. Consider the case where we have c equality constraints. Then A has c rows, and $A\Sigma^{-1}A'$ is dimensioned c by c. The number of equality constraints will typically be much smaller than n, and as such $A\Sigma^{-1}A'$ will be fast to invert.

This fast analytic approach is a good starting point for a backtest. It guarantees the optimal portfolio at each time point - albeit the optimal portfolio for the equality constrained optimization. Thus, it provides the 'best' portfolio with the correct sum-to conditions (typically either fully invested or zero invested), and can also include equality constrained stock or sector weights. However, it cannot include inequality constraints such as maximum or minimum stock or sector weights.

Once Σ^{-1} has been determined (which we show how to do efficiently in Section 3), Equation (2) only requires a few matrix multiplications and additions and thus can be updated very quickly.

However, this approach has limitations from a practical point of view. It may produce solutions where stocks have impractically large positive or negative weightings. Assuming that the return expectations are constrained to within reasonable levels, and that we disallow unreasonably low correlations (close to minus one), then typically these outlier weightings will happen in cases where the expected covariances of those stocks are unreasonably low.

Such extreme weights may bias the backtesting results - if in some time period, the portfolio just happened to have an extreme weight in a stock, and that stock just happened to have a high or low return for the same period, then the overall performance may end up being unrepresentative of the true value of the strategy being tested.

However, if sufficient care is taken to ensure that no unreasonably low variances are estimated for any stock using methods such as those of Ledoit and Wolf (2004) and its references, then these cases can largely be avoided, and the analytic approach can be used as a useful first pass for the identification of promising investment approaches.

Indeed, once this first pass is completed, we propose using more accurate methods on the promising cases to further verify their efficacy. We introduce one such approach in the next subsection.

2.2. *Iterated analytic approach*

It is possible to overcome the overweight/underweight limitations of the previous subsection by imposing a constraint set with appropriate minimum and maximum stock weights. For instance, this could be set to match the real world (long only) set up of not allowing shorting, and also of not putting too much weight into any one stock.

Unfortunately, mean-variance optimization tasks with inequality constraints are not analytically soluble. Typically, iterative procedures based on some measure of the gradient of the optimization function are used. See for instance Kelley (1999). As stated in the introduction, such techniques may be infeasibly slow for our purpose.

Instead, we add a heuristic iterative approach to our equality constraint optimization to estimate the portfolios in a very computationally (and hence time) efficient manner. This gives a good approximation to the optimal portfolios, and is still very fast compared to the standard approaches.

The idea is to avoid the costly task of optimising an inequality constrained setting. Instead, we modify the optimization task so that the equality constrained solution obeys all inequality constraints naturally.

If we exclude inequality constraints, then a mean variance optimization problem has four elements: the covariance matrix, the expected returns, the equality constraints and the risk aversion parameter.

Thus, the problem can be modified in any of those four ways, or in combination. We advocate and test the modification of the covariance matrix for the following reasons:

- **Covariance matrix adjustment**
  Our techniques are aimed at restraining breaches of non-negative upper bounds and non-positive lower bounds. In this case, we will always be increasing estimated variances. This is a very conservative approach - solving an optimization for the case where some of the stocks are deemed to be more risky than measured. Ultimately, our portfolio will always be optimal for a case similar to the 'full' problem, but with some of the stocks being deemed to have higher than measured variances.
- **Expected return adjustment**
  Without complicated and time-intensive look-back provisions in our iteration, there is some danger that a stock whose return estimate was lowered in one iteration, may via changes in another stocks' returns from a later iteration, end up with a negative weight due to the initial change in its estimated return. Whilst similar events are theoretically possible with the update to variance method, we see this as a higher risk. Addition-

ally, the approach in this paper is designed to be agnostic to the source of the return estimates. Given that we have no knowledge of how they were generated, or of their characteristics, we do not seek to modify them.

- **Adding equality constraints**
  Another approach would be to add equality constraints to stocks outside of their limits which force them back inside the limits. The problem with this approach is that the iteration can quickly generate so many conflicting constraints that any solution at all is infeasible.
- **Changing the risk aversion parameter**
  Changing the risk parameter is equivalent to multiplying the covariance matrix by a scalar, and hence is simply a subset of what is possible via covariance matrix adjustment.

The iterative heuristic is detailed below. Since the analytic solution to the mean-variance optimization problem uses the inverse of the covariance matrix rather than the covariance matrix, in order to minimise computational effort we express the heuristic in terms of the inverse of the covariance matrix.

Heuristic 2.2. *Equality and Inequality Constrained Optimization:*

Consider the following constrained portfolio optimization task:

$$\arg\max_{\mathbf{w}} \left\{\mathbf{w}'\mathbf{r} - \tfrac{\lambda}{2}\mathbf{w}'\Sigma\mathbf{w}\right\}$$
$$\text{subject to} A\mathbf{w} = \mathbf{b};\ \ \mathbf{w} \geq \mathbf{w_l};\ \ \mathbf{w} \leq \mathbf{w_m}$$
$$\text{where } \mathbf{w_l} \leq 0;\ \ \mathbf{w_m} \geq 0$$

1. Begin with a copy of Σ^{-1}. Denote this Σ^{*-1}
2. Do an analytical optimization using Σ^{*-1}, using Lemma 2.1..
3. If all of the weights are within their upper and lower bounds, then stop. (Otherwise continue.)
4. If the cumulative time taken for this heuristic is more than 25 percent of that of a standard optimization then perform a standard optimization, and stop.
5. For any stocks outside of non-zero lower or upper bounds:
   - Multiply Σ^{*-1} by 0.95 for the rows of such stocks.
   - Multiply Σ^{*-1} by 0.95 for the columns of such stocks.
6. For stocks outside of zero bounds:
   - Set Σ^{*-1} to zero for the rows and columns of these stocks.
7. Repeat from Step 2.

The logic of Heuristic (2.2) is as follows. Multiplying the *nth* column and then the *nth* row of Σ^{-1} by a constant is equivalent to pre and post multiplying Σ^{-1} by an identity matrix with the *nth* diagonal element replaced by the square root of the constant. This is exactly equivalent to pre and post multiplying Σ by an identity matrix with the *nth* diagonal element replaced by the reciprocal of the square root of the constant. In other words, keeping the same correlations between variables (stocks) but increasing the variance of the *nth* stock by that factor.

Thus, on each iteration where we multiply Σ^{-1} by 0.95 for certain rows and columns, we are effectively increasing the variance on the stocks represented by those rows and columns by a little over 5 percent.

Also, on each heuristic where we multiply Σ^{-1} by 0, we are effectively setting those stocks to have infinite variance, and thus for a risk penalised optimization they cannot be held at anything but zero weight. This is appropriate for stocks which need to be set to zero weight.

Tests of Heuristic (2.2) with both empirical and synthetic (random) data have let to the following observations:

- It works best when few boundaries are hit in the first optimization. Indeed, if no boundaries are hit then it needs only one iteration and supplies the exact solution to the inequality constrained optimization.
- Although it converged in the vast majority of cases, we were able to identify a small proportion of pathological cases where it either did not converge or converged so slowly as to deem the approach useless. That is why we put the cut-off of 25 percent of the time of a standard optimization. These cases were sufficiently rare as to make the average cost of this escape negligible, whilst guaranteeing a solution every time.
- The factor of 0.95 was found to be a good trade-off between quickly getting to a point where constraints were satisfied versus the risk of overshooting the boundaries and providing overly conservative weights. This factor can, of course, be tuned to be appropriate for the particular problem set up and data characteristics.

Tests of the heuristic were performed on historical stock data. The results of these tests are written up in Section 4.

3. Covariance and inverse covariance estimation

For analytical optimization, we need the inverse of the covariance matrix. Calculating this at each time step via a complete re-estimation of the covariance matix and then performing a matrix inverse is computationally expensive.

In this section, we introduce a computationally efficient update method for the inverse of the covariance matrix. Whilst such methods are well-known in engineering, they are less well-known in finance.

Furthermore, in engineering the applications typically involve a fixed number of variables. However in finance, the number of variables (stocks) changes over time as stocks are added into and out of the index, are delisted, etc. The novel techniques of this section are an extension of the recursive techniques to the situation of changing variables.

Also note that while we don't explicitly apply procedures to minimise estimation error in the covariance matrix, such as those from Ledoit and Wolf (2004), these can easily be incorporated into our efficient methodologies.

3.1. Recursive Covariance Estimation

There are well known methods for recursive covariance estimation, and here we re-derive one as a starting point for our analysis.

Lemma 3.1. *Define an exponentially smoothed (increasing window length) covariance estimate at time k as* Σ_k. α *is the smoothing parameter where* $0 < \alpha < 1$. $\boldsymbol{\mu}_k$ *is the exponentially weighted average up to time k. Then we have a recursive formula for updating this estimate at each time step:*

$$\Sigma_k = \left(\frac{\alpha^{k+1}-\alpha}{\alpha^{k+1}-1}\right)\Sigma_{k-1} + \frac{\alpha-1}{\alpha(\alpha^k-1)}(\mathbf{x}_k-\boldsymbol{\mu}_k)(\mathbf{x}_k-\boldsymbol{\mu}_k)' \quad (3)$$

$$= \left(\frac{\alpha^{k+1}-\alpha}{\alpha^{k+1}-1}\right)\Sigma_{k-1} + \frac{(\alpha-1)(1-\frac{\alpha-1}{\alpha^{k+1}-1})}{\alpha^{k+1}-1} (\mathbf{x}_k-\boldsymbol{\mu}_{k-1})(\mathbf{x}_k-\boldsymbol{\mu}_{k-1})' \quad (4)$$

Furthermore, in the limit as $k \to \infty$ *we have*

$$\lim_{k\to\infty}\Sigma_k \approx \alpha\Sigma_{k-1} + (\frac{1}{\alpha}-1)(\mathbf{x}_k-\boldsymbol{\mu}_k)(\mathbf{x}_k-\boldsymbol{\mu}_k)' \approx \alpha\Sigma_{k-1} + \alpha(1-\alpha)(\mathbf{x}_k-\boldsymbol{\mu}_{k-1})(\mathbf{x}_k-\boldsymbol{\mu}_{k-1})'$$

The proof can be seen in the appendix.

Lemma 3.1. provides an efficient means of updating the covariance matrix for a fixed set of variables (stocks) as we move through time. The added task of adding or removing variables (stocks) is as simple as appending or removing rows and columns. By combining this with the recursive approach above, we have a relatively computationally efficient method of updating the covariance matrix even in the case of changing variables.

However, on those occasions where stocks (variables) are changed, then the calculation of the new covariances can be slow for long data sets, as we cannot use recursive techniques for variables we haven't previously calculated. To further speed up the process, we also introduce a fixed window length version of the recursion formula. Setting an appropriate window length (determined by the decay factor), we can then achieve both an accurate estimation of the covariance matrix and even faster computation.

Lemma 3.2. *Define an exponentially smoothed covariance estimate with a fixed window length of n at time k as* Σ_k^n. α *is the smoothing parameter where* $0 < \alpha < 1$. $\boldsymbol{\mu}_k$ *is the exponentially weighted average up to time k. Then we have a recursive formula for updating this estimate at each time step:*

$$\Sigma_k^n = \alpha\Sigma_{k-1}^n + \alpha d_k d_k' + \frac{\alpha-1}{\alpha^n-1}(\mathbf{x_k}-\boldsymbol{\mu_k})(\mathbf{x_k}-\boldsymbol{\mu_k})' - \frac{\alpha^n(\alpha-1)}{\alpha^n-1}(\mathbf{x}_{k-n}-\boldsymbol{\mu}_k)(\mathbf{x}_{k-n}-\boldsymbol{\mu}_k)' \quad (5)$$

Where $d_k \underline{\underline{\text{def}}} \quad \boldsymbol{\mu}_k - \boldsymbol{\mu}_{k-1} = \boldsymbol{\mu}_k(1-\frac{1}{\alpha}) + \frac{\alpha-1}{\alpha^{n+1}-\alpha}[\mathbf{x}_k - \alpha^n\mathbf{x}_{k-n}]$.

Furthermore, in the case where α^n is small, we have the following approximation:

$$\Sigma_k^n \approx \alpha \Sigma_{k-1}^n + \left(\frac{1}{\alpha} - 1\right)(\mathbf{x}_k - \boldsymbol{\mu}_k)(\mathbf{x}_k - \boldsymbol{\mu}_k)' \quad (6)$$

The proof can be seen in the appendix.

For the case where the constituents of the universe, or equivalently the variables in the covariance matrix, are being changed, it is simple to update the covariance matrix. To delete variables we can simply remove the rows and columns representing those variables. To add variables, we can work out the variance of each new variable and also its covariance with each of the other variables, and augment the covariance matrix accordingly.

Thus we have available computationally efficient (fast) techniques for the estimation of the covariance matrix. However, the analytic and iterative analytic responses we have listed above require the inverse of the covariance matrix, and inverting the covariance matrix is computationally expensive.

In the next subsection, we introduce computationally efficient time and variable update methods for the inverse of the covariance matrix.

3.2. Recursive estimation of the inverse of the covariance matrix

The Inverse Covariance Matrix (ICM) can also be efficiently updated. First we derive formulae for the simple case where no variables (stocks) are being added or removed. Then we derive formulae for the case where stocks are added, and finally we derive formulae for the case where stocks are removed.

The general case is simply achieved by sequentially using all three methods we develop - first updating the inverse covariance matrix for fixed stocks; then adding new ones; then removing old ones.

3.2.1. Updating the inverse of the covariance matrix with fixed variables (stocks)

Lemma 3.3. *Consider the case of updating the inverse of the covariance matrix for the increasing window length, exponential smoothing covariance estimation. Define a and b as the coefficients in (3). That is,* $a \stackrel{\text{def}}{=} \frac{\alpha^{k+1}-\alpha}{\alpha^{k+1}-1}$ *and* $b \stackrel{\text{def}}{=} \frac{\alpha-1}{\alpha(\alpha^k-1)}$. *Then we have:*

$$\Sigma_k^{-1} = a^{-1}\Sigma_{k-1}^{-1} - \frac{b\Sigma_{k-1}^{-1}(\mathbf{x}_k - \boldsymbol{\mu}_k)(\mathbf{x}_k - \boldsymbol{\mu}_k)'\Sigma_{k-1}^{-1}}{a^2 + ab(\mathbf{x}_k - \boldsymbol{\mu}_k)'\Sigma_{k-1}^{-1}(\mathbf{x}_k - \boldsymbol{\mu}_k)} \quad (7)$$

The proof can be seen in the appendix.

Lemma 3.3 offers an efficient method to update the inverse of the covariance matrix. Note that the denominator of the right hand term is a scalar, and as such the calculation involves only a few matrix multiplications and additions, and two scalar divisions.

The extent of the speed up is large. For instance, we ran a simulation using Python and Numpy for 100 stocks with random data representing the stock returns. The calculation of 300 updated covariance matrices took 3.7 seconds using the recursive method, but 400.5 seconds using direct calculations. The speed ratio was thus 107. The matrix norm of the difference between the final inverse covariance matrix using each method divided by the matrix norm of the fully calculated final inverse covariance matrix was 5E-11, indicating that no numerical issues arose in the calculation.

However, the issue with the above update is that the window length is increasing without limit. This can add computational burden, especially in the case where new stocks are added. To this end, we also introduce the following lemma for the fixed window length case:

Lemma 3.4. *Consider the case of updating the inverse of the covariance matrix for the fixed window length of n, exponential smoothing covariance estimation. Assume furthermore that* α^n *is small. Then we have:*

$$\Sigma_k^{-1} \approx \alpha^{-1}\Sigma_{k-1}^{-1} - \frac{(\frac{1}{\alpha} - 1)\Sigma_{k-1}^{-1}(\mathbf{x}_k - \boldsymbol{\mu}_k)(\mathbf{x}_k - \boldsymbol{\mu}_k)'\Sigma_{k-1}^{-1}}{\alpha^2 + (1-\alpha)(\mathbf{x}_k - \boldsymbol{\mu}_k)'\Sigma_{k-1}^{-1}(\mathbf{x}_k - \boldsymbol{\mu}_k)} \quad (8)$$

Lemmas 3.3 and 3.4 would be sufficient in the case of fixed variables (stocks). However, we also require the ability to efficiently update the inverse of the covariance matrix where stocks are added or removed from the universe.

3.2.2. Updating the inverse of the covariance matrix with new stocks added

Consider the case where we have new stocks added. We assume that our original stock universe contained n stocks, and we are adding $m << n$ stocks. This represents the case of a typical index change where the index may have hundreds of constituents, and at an index change only a few are added.

Denote the matrices of returns to the original n stocks as R_O, for the new m stocks as R_N, and for the aggregated set as $R_A \stackrel{\text{def}}{=} [R_O \; R_N]$. Here each row

represents the return over one time period, and each column represents one security.

Using methods of the previous section, we can quickly update the covariance of the original stocks, Σ_O, and of the new stocks Σ_N, and of the aggregated set of stocks, Σ_A. For clarity, we define the blocks of the aggregated covariance matrix as:

$$\Sigma_A = \begin{bmatrix} \Sigma_O & \Sigma_{ON} \\ \Sigma'_{ON} & \Sigma_N \end{bmatrix} \tag{9}$$

Our task is to efficiently calculate the inverse of this matrix. Since we have already efficiently calculated the inverse of the covariance of the original stocks from Lemma 3.3, we can use this result and the Matrix Inversion Lemma (see Henderson and Searle (1981)) to achieve this goal.

Lemma 3.5. *Assume we know the inverse of the covariance matrix Σ_O for a set of n 'original' stocks with returns R_O, and we wish to efficiently calculate the inverse of the augmented covariance matrix Σ_A which includes an additional m 'new' stocks with returns R_N. Define Σ_{ON} as the covariance between the original n stocks and the augmented m stocks, or equivalently the upper right n by m block of the aggregated covariance matrix Σ_A. Also define the (small) m by m matrix as $\Theta \underline{\underline{\text{def}}} (\Sigma_N - \Sigma'_{ON}\Sigma_O^{-1}\Sigma_{ON})^{-1}$ and the (large) n by n matrix as $\Lambda \underline{\underline{\text{def}}} \Sigma_O^{-1} + \Sigma_O^{-1}\Sigma_{ON}\Theta\Sigma'_{ON}\Sigma_O^{-1}$. Then we have*

$$\Sigma_A^{-1} = \begin{bmatrix} \Lambda & -\Sigma_O^{-1}\Sigma_{ON}\Theta \\ -\Sigma_N^{-1}\Sigma'_{ON}\Lambda & \Theta \end{bmatrix} \tag{10}$$

The proof can be seen in the appendix.

Lemma 3.5 avoids the costly inversion of an $n + m$ by $n + m$ matrix, and instead only requires the inversion of two m by m matrices, and a few matrix additions and multiplications.

We already know the inverse of the original set of stocks Σ_O, efficiently calculated via Lemma 3.3. The only new inverses required here are for Θ and Σ_N, both of which are small, dimensioned m by m, and which thus can be calculated quickly.

Simulations of calculating an inverse covariance matrix made up of 200 stocks (for which the covariance matrix is previously known), and adding 10 new stocks shows that the method of Lemma 3.5 is in this case approximately 4 times faster than the naive method requiring inversion of the full 210 stock augmented covariance matrix.

3.2.3. Updating the inverse of the covariance matrix with old stocks deleted

Consider the case where stocks have dropped out of either the index or the investable universe. Here we have the inverse of a larger covariance matrix and wish to calculate the inverse of a smaller covariance matrix, one which excludes the dropped stocks.

Assume in the first instance that the columns to be deleted are at the far right of the matrix, and the rows to be deleted are at the bottom of the matrix. We start with $n + m$ stocks, and drop the rightmost and lowest m.

Lemma 3.6. *Assume we know the inverse of the augmented covariance matrix Σ_A for a set of $n + m$ 'continuing' and 'to be deleted' stocks, and we wish to efficiently calculate the inverse of the upper leftmost n by n block of the covariance matrix (the continuing stocks), denoted Σ_O. This represents the inverse of the covariance matrix of the stocks excluding those deleted. Denote Λ as the top left n by n submatrix of Σ_A^{-1}, Θ as the bottom right m by m submatrix of Σ_A^{-1}, and Φ as the top right n by m submatrix of Σ_A^{-1}. Then we have:*

$$\Sigma_O^{-1} = \Lambda - \Phi\Theta^{-1}\Phi' \tag{11}$$

The proof can be seen in the appendix.

The only inversion required here is that of Θ which being m by m is fast, assuming a typical index change case of $m << n$.

Simulations where we started with a known inverse of a 210 stock covariance matrix, and required the inverse of the covariance matrix of the first 200 stocks show that the method of Lemma 3.6. is in this case approximately 10 times faster than the naive method requiring inversion of the full 200 stock covariance matrix.

If the stocks to be deleted are not the right most stocks of the covariance, then it is straightforward to manipulate the covariance matrix into the correct order.

Lemma 3.7. *Consider a covariance matrix, Σ for a given set of variables (stocks) x, and a permutation matrix P. Then the covariance matrix Σ_P for the permuted variables xP and its' inverse can be expressed as:*

$$\Sigma_P = P'\Sigma P \tag{12}$$

$$\Sigma_P^{-1} = P'\Sigma^{-1}P \tag{13}$$

The proof can be seen in the appendix.

Effectively, Equations (12) and (13) state that a permutation of stocks in the covariance matrix is equivalent to an indentical permutation of stocks in the inverse of the covariance matrix, both arising from a permutation of the order of stocks.

This offers a fast method to get all of the stocks which are to be deleted to the rightmost side of the covariance matrix, thus permitting the use of Lemma 3.6. for the general case of the deletion of any stocks.

3.2.4. General case - simultaneous addition and deletion of stocks and updating the inverse of the covariance matrix

In the general case, it is trivial to apply the results of Lemmas 3.3, 3.5, and 3.6 sequentially - first updating the inverse covariance matrix for the existing stocks; then adding and finally deleting any changes to the investible universe of stocks.

Thus we have a computationally efficient, recursive solution for the calculation of the inverse of the covariance matrix.

4. Empirical example

To explore the utility of our approach in a practical setting, we analyse candidate investment strategies in the United States market. The data is sourced from Citi Equities, and consists of daily capitalisation, price and total return data over a five year period from January 2008 to December 2012. The universe consisted of the top 150 stocks by market capitalisation. Changes to the index were allowed to occur once per five trading days. Additionally, to filter out bad data, stock returns were culled at 50 percent per day.

The goal of the methods developed in this paper is to provide a fast comparison of investment strategies, from which the most appealing can be selected for a slower but more comprehensive analysis. To this end, we created four synthetic prediction sets. The first set had a negative correlation to the true returns, and each successive set had a higher correlation to the true returns.

The objective of the backtest was to determine whether the fast methods would correctly rank the prediction sets on the basis of portfolio information ratios.

To this end, we re-ran the analysis with different combinations of optimization and covariance estimation methods. We wanted to determine whether the fast methods gave the same results as the slower methods, and to measure the speed increment.

The optimization methods used were:

- Analytic Optimization
  This used the results of Lemma 2.1, Equation (2). Note that while this method gives an 'optimal' (here sum-to-zero) solution for the unconstrained case, it is risks having unreasonably large positions in particular stocks as it doesn't offer any restriction on maximum overweights and underweights.
- Iterated Analytic Optimization
  Here we used Heuristic 2.2. This method satisfies both the sum-to constraint (sum-to-zero in this example), as well as the stock maximum and minimum active weight constraints.
- Full Optimization
  A standard Mean-Variance optimization was carried out using the QP routine from CVXOPT, Dahl and Vandenberghe (2008). This method satisfies all constraints.

The covariance methods used were:

- Full Covariance
  This used standard techniques where the full covariance was estimated at each time step, and its inverse calculated where necessary.
- Recursive Covariance
  Here we used the results of Section 3 to recursively calculate the covariance matrix at each time step based on previous results. Note that to prevent any long-term drift, each 100 time steps we re-estimated the covariance and inverse covariance matrices using a full calculation. This had minimal impact on speed as it only occurred infrequently, but this periodic resetting helped eliminate drift due to inexact numerical computing.
- Cached Covariance
  In cases where the stock universe is known, and the only iteration is over the alphas (expected returns), then it is simple to store all of the covariance or inverse covariance matrices on the first pass. This eliminates time spent on covariance estimation from all subsequent passes. We only advocate the non-cached recursive estimation procedures for cases where caching is infeasible such as real-time adaptive trading strategies.

For brevity, we do not report all 9 combinations. We compare the standard approach (full covariance and full optimization) with two faster methods (recursive covariance and analytic optimization) and (recursive covariance and iterative optimization).

We chose these combinations because they are the most practical implementations. It wouldn't make sense, for instance, to use the slow full optimization with the fast covariance estimation as the relative speed up wouldn't justify the approximation.

Additionally, we don't list the results with the cached covariance as they will be identical to those with the fully calculated covariance, except of course to be much faster.

We show the speed of the various methods in Tables 1, 2 and 3.

The results indicate the effectiveness of these techniques for this example:

- The recursive covariance method in this case is 5 times faster than the standard method. When we used cached covariance storage, it became 130 times faster.
- The iterative optimization is 6 times as fast as the full optimization, and the analytic version is 152 times as fast.
- The total backtest includes a lot of other tasks which are common to all optimization/covariance estimation versions and therefore do not speed up. Even so, the fastest version is thirty eight times faster than the standard methods!

Table 1
Comparison of Covariance times under different methods

Covariance calculation method	Time Taken (s)
Cached	0.1
Recursive	3.1
Full	16.9

Table 2
Comparison of Optimization times under different methods

Optimization method	Time taken (s)
Analytic	0.9
Iterative	23.6
Full	140.9

Table 3
Comparison of Total Backtest times under different methods

Covariance/Optimization methods	Time taken (s)
Cached/Analytic	4.3
Recursive/Iterative	30.2
Full/Full	164.1

However, for our approach to be useful, speed is not enough. It has to also correctly order the various strategies in terms of usefulness. Once it has done so, the most accurate approaches can be used to further tune the most promising approaches.

Figure 1 shows the information ratio (average active return divided by standard deviation of active return) for each of the different strategies, grouped across different estimation/optimization methods.

What is clear, is that the ordering of the strategies is identical. So, in this example, the fastest method accurately points correctly to the best set of predictions.

This indicates that, for our example, it makes sense to use the faster methods to identify the best candidate strategies, with further work possible using more accurate methods once the best candidates have been identified.

Of course, in this toy example of only four different predictive models, it is fine to use even the slowest methods. However, in more complex examples, for instance quantitative methods with many parameters to tune, then the faster methods may become invaluable.

In practice, the efficacy and accuracy of the fast methods will depend strongly on a number of factors. For instance, above about 200 stocks we found that the accuracy lowers, however the speed improvement typically increases. For this reason, the strategy must be tuned appropriately for each particular task.

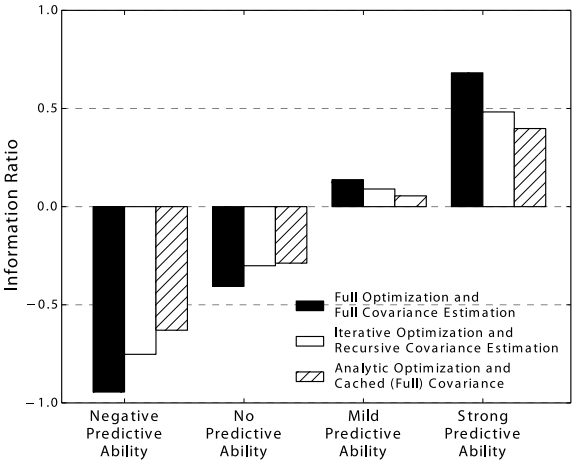

Fig. 1. Momentum Length Analysis via different Estimation and Optimization Methods.

5. Conclusions

We have developed extremely fast techniques for risk (covariance) estimation and for mean-variance (quadratic) portfolio optimization. This speed gives a portfolio manager the ability to quickly test a much wider range of strategies than previously possible.

These techniques are particularly applicable to the task of identifying and optimising systematic investment approaches where useful signals from among many parameterized candidates need to be identified and optimized.

The speed does come at a cost of approximation, and for this reason we propose a fast first cut analysis to determine which ranges of parameter space are most promising, followed by a more accurate and more detailed analysis using slower but exact standard methods within such ranges.

Furthermore, since standard methods estimate the covariance matrix from scratch at each time period, and then apply an essentially black-box optimization procedure, they make it impossible to directly see how changes in the data (stock returns) lead to changes in the optimal portfolio.

An interesting byproduct of our approach is the ability to analytically determine directly how new stock return data will impact the optimal portfolio via perturbations to the covariance matrix, possibly coupled with changes to the return estimates. This will form the basis of future research.

Acknowlegements

The author wishes to express his appreciation to IFM Investors, and in particular the Indexed Equities Team for supporting this research.

References

Amenc, N., Goltz, F., Martellini, L., Retkowsky, P., 2010. Effcient indexation: An alternative to cap-weighted indices. An EDHEC-Risk Institute Publication.

Anderson, B.D., Moore, J.B., 1979. Optimal filtering, volume 11. Prentice-hall Englewood Cliffs, NJ.

Arnott, R.D., Hsu, J.C., Moore, P., 2005. Fundamental indexation. Financial Analysts Journal. 61.

Best, M.J., 2010. Portfolio Optimization. Chapman & Hall / CRC. ISBN 978-1-4200-8584-6.

Brown, B.R., 2010. Chasing the Same Signals: How Black-Box Trading Influences Stock Markets from Wall Street to Shanghai. Wiley Trading. John Wiley & Sons. ISBN 9780470824887.

Chan, L.K., Jegadish, N., Lakonishok, J., 1996. Momentum strategies. The Journal of Finance. 51(5), 1681–1713. ISSN 1540-6261.

Dahl, J., Vandenberghe, L., 2008. Cvxopt – a python package for convex optimization. URL http://cvxopt.org/.

Do, B., Faff, R., 2010. Does simple pairs trading still work? Financial Analysts Journal. 66(4), 83.

Fama, E.F., French, K.R., 1992. The cross-section of expected stock returns. The Journal of Finance. 157(2).

Fama, E.F., French, K.R., 2012. Size, value, and momentum in international stock returns. Journal of Financial Economics. 105(3), 457–472. ISSN 0304-405X.

Figlewski, S., Gao, B., 1999. The adaptive mesh model: A new approach to effcient option pricing. Journal of Financial Economics. 53.

Goodwin, T.H., 1998. The information ratio. Financial Analysts Journal. pp. 34–43.

Haugen, R.A., Baker, N.L., 1991. The effcient market ineffciency of capitalization-weighted stock portfolios. The journal of portfolio management. 17(3), 35–40.

Haugen, R.A., Baker, N.L., 2010. Case closed. Handbook of Portfolio Construction, pp. 601–619.

Henderson, H.V., Searle, S.R., 1981. On deriving the inverse of a sum of matrices. SIAM Review. 23(1).

Kelley, T., 1999. Iterative Methods for Optimization. Society for Industrial and Applied Mathematics.

Kritzman, M., 1998. How to detect skill in management performance. Street-wise: The Best of The Journal of Portfolio Management, page 101.

Ledoit, O., Wolf, M., 2004. Honey, I shrunk the sample covariance matrix. Journal of Portfolio Management. 30(4).

Liu, M., Liu, Q., Ma, T., 2011. The 52-week high momentum strategy in international stock markets. Journal of International Money and Finance. 30(1), 180–204.

Poterba, J.M., Summers, L.H., 1988. Mean reversion in stock prices: Evidence and implications. Journal of Financial Economics. 22(1), 27–59.

Rachev, S., Jašić, T., Stoyanov, S., Fabozzi, F.J., 2007. Momentum strategies based on reward–risk stock selection criteria. Journal of Banking & Finance. 31(8), 2325–2346.

Scherer, B., 2011. A note on the returns from minimum variance investing. Journal of Empirical Finance. 18(4), 652–660.

Schulmeister, S., 2009. Profitability of technical stock trading: Has it moved from daily to intraday data? Review of Financial Economics, 18(4), 190–201.

Viveros-Jimnez, F., Len-Borges, J.A., Cruz-Corts, N., 2013. A simple adaptive algorithm for numerical optimization. In Ildar Batyr-shin and Miguel, G.M., editors. Advances in Computational Intelligence, volume 7630 of Lecture Notes in Computer Science, pages 115–126. Springer Berlin Heidelberg. ISBN 978-3-642-37797-6.

Young, P.C. 2011. Recursive Estimation and Time-Series Analysis: An Intro-duction for the Student and Practitioner. Springer.

$$\begin{bmatrix} \lambda\Sigma A' \\ A \;\; 0 \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{\lambda}\Sigma^{-1}\left[I - A'\left(A\Sigma^{-1}A'\right)^{-1}A\Sigma^{-1}\right] & \Sigma^{-1}A'\left(A\Sigma^{-1}A'\right)^{-1} \\ \left(A\Sigma^{-1}A'\right)^{-1}A\Sigma^{-1} & -\lambda\left(A\Sigma^{-1}A'\right)^{-1} \end{bmatrix} \underline{\underline{\text{def}}} \begin{bmatrix} \frac{1}{\lambda}\Sigma^{-1}\left[I - A'\Phi A\Sigma^{-1}\right] & \Sigma^{-1}A'\Phi \\ \Phi A\Sigma^{-1} & -\lambda\Phi \end{bmatrix} \tag{16}$$

Appendix

Proof of Lemma 2.1.

The first step is to find an analytical expression for the optimal portfolio weights, w^*, where these weights are the solution of Equation (14).

$$\mathbf{w}^* = \arg max_w \left\{ \mathbf{w}'\mathbf{r} - \frac{\lambda}{2}\mathbf{w}'\Sigma\mathbf{w} \;\mid\; A\mathbf{w} = \mathbf{b} \right\} \tag{14}$$

This solution is well known, see for instance Theorem 1.3 within Best (2010). In our notation the solution can be expressed in matrix form as:

$$\begin{bmatrix} \lambda\Sigma \;\; A' \\ A \;\; 0 \end{bmatrix} \begin{bmatrix} \mathbf{w}^* \\ \mathbf{g} \end{bmatrix} = \begin{bmatrix} \mathbf{r} \\ \mathbf{b} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{w}^* \\ \mathbf{g} \end{bmatrix} = \begin{bmatrix} \lambda\Sigma \;\; A' \\ A \;\; 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{r} \\ \mathbf{b} \end{bmatrix} \tag{15}$$

This can then be calculated easily via the block matrix inversion formula of (24) as follows:

Where $\Phi \underline{\underline{\text{def}}} \left(A\Sigma^{-1}A'\right)^{-1}$. Applying (16) to (15) then proves Equation (1).

Equation (2) is derived by specialising Equation (1) to the case where $A = \mathbf{1}'$, and b is a scalar. Then $A'A\Sigma^{-1}\mathbf{r} = \mathbf{1}(\mathbf{1}'\Sigma^{-1}\mathbf{r})$ and $A\Sigma^{-1}\mathbf{r} = \mathbf{1}'\Sigma^{-1}\mathbf{r}$ which is a scalar and therefore commutes with matrices. Substituting these into Equation (1) then gives Equation (2). ■

Proof of Lemma 3.1.

Define $\beta_k = \frac{1}{\sum_{i=0}^{k}\alpha^{k-i}}$ and Σ_k as the weighted estimate of the covariance matrix estimated at time k, and $\boldsymbol{\mu}_k = \beta_k \sum_{i=1}^{k} \mathbf{x}_i\alpha^{k-i}$ as the weighted estimate of the average of $\mathbf{x}$, estimated at time k. Also we have a difference operator, $\mathbf{d}_k \underline{\underline{\text{def}}} \; \boldsymbol{\mu}_k - \boldsymbol{\mu}_{k-1}$.

Then, for the case of exponential smoothing, we have:

$$\begin{aligned}
\Sigma_k &= \beta_k \sum_{i=0}^{k} \alpha^{k-i}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)' \\
&= \alpha\beta_k \sum_{i=0}^{k-1} \alpha^{k-1-i}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)' + \beta_k(\mathbf{x}_k - \boldsymbol{\mu}_k)(\mathbf{x}_k - \boldsymbol{\mu}_k)' \\
&= \alpha\beta_k \sum_{i=0}^{k-1} \alpha^{k-1-i}(\mathbf{x}_i - \boldsymbol{\mu}_{k-1} - \mathbf{d}_k)(\mathbf{x}_i - \boldsymbol{\mu}_{k-1} - \mathbf{d}_k)' + \beta_k(\mathbf{x}_k - \boldsymbol{\mu}_k)(\mathbf{x}_k - \boldsymbol{\mu}_k)' \\
&= \alpha\beta_k \sum_{i=0}^{k-1} \alpha^{k-1-i} \begin{bmatrix} (\mathbf{x}_i - \boldsymbol{\mu}_{k-1})(\mathbf{x}_i - \boldsymbol{\mu}_{k-1})' \\ +(\boldsymbol{\mu}_{k-1} - \mathbf{x}_i)\mathbf{d}_k' + \mathbf{d}_k(\boldsymbol{\mu}_{k-1} - \mathbf{x}_i)' \\ +\mathbf{d}_k\mathbf{d}_k' \end{bmatrix} + \beta_k(\mathbf{x}_k - \boldsymbol{\mu}_k)(\mathbf{x}_k - \boldsymbol{\mu}_k)' \\
&= \alpha\frac{\beta_k}{\beta_{k-1}}\Sigma_{k-1} + \beta_k\left(\frac{\alpha}{\beta_{k-1}}\mathbf{d}_k\mathbf{d}_k' + (\mathbf{x}_k - \boldsymbol{\mu}_k)(\mathbf{x}_k - \boldsymbol{\mu}_k)'\right) \\
&= \alpha\frac{\beta_k}{\beta_{k-1}}\Sigma_{k-1} + \beta_k\left(\frac{\beta_{k-1}}{\alpha} + 1\right)(\mathbf{x}_k - \boldsymbol{\mu}_k)(\mathbf{x}_k - \boldsymbol{\mu}_k)'
\end{aligned}$$

The second last line follows since:

$$\begin{aligned}\sum_{i=0}^{k-1}\alpha^{k-1-i}(\boldsymbol{\mu}_{k-1}-\mathbf{x}_i) &= \boldsymbol{\mu}_{k-1}\sum_{i=0}^{n-1}\alpha^{k-1-i} \\ &\quad -\sum_{i=0}^{k-1}\alpha^{k-1-i}\mathbf{x}_i \\ &= \frac{\boldsymbol{\mu}_{k-1}}{\beta_{k-1}}-\frac{\boldsymbol{\mu}_{k-1}}{\beta_{k-1}}=0.\end{aligned}$$

The last line follows since:

$$\begin{aligned}\mathbf{d}_k &= \boldsymbol{\mu}_k-\boldsymbol{\mu}_{k-1} \\ &= \boldsymbol{\mu}_k-\frac{\beta_{k-1}}{\alpha}\left[\sum_{i=0}^{k}\alpha^{k-i}\mathbf{x}_i-\mathbf{x}_k\right] \\ &= \boldsymbol{\mu}_k-\frac{\beta_{k-1}}{\alpha\beta_k}\boldsymbol{\mu}_k+\frac{\beta_{k-1}}{\alpha}\mathbf{x}_k \\ &= \frac{\beta_{k-1}}{\alpha}\left(\mathbf{x}_k+\boldsymbol{\mu}_k(\frac{\alpha}{\beta_{k-1}}-\frac{1}{\beta_k})\right) \\ &= \frac{\beta_{k-1}}{\alpha}\left(\mathbf{x}_k-\boldsymbol{\mu}_k\right) \qquad (17)\end{aligned}$$

Note also that expansion of the formulae for β_k and β_{k-1} lead to the identity $\frac{\alpha}{\beta_{k-1}}-\frac{1}{\beta_k}=-1$.

Then applying the geometric series summation formula for β^k and β^{k-1} provides (3).

We derive (4) which is similar to (3) but based on $(\mathbf{x}_k-\boldsymbol{\mu}_{k-1})$ in place of $(\mathbf{x}_k-\boldsymbol{\mu}_k)$.

First note that $\mathbf{d}_k=\beta_k\sum_{i=0}^{k}\alpha^{k-i}\mathbf{x}_i-\boldsymbol{\mu}_{k-1}=\beta_k\mathbf{x}_k+\frac{\beta_k\alpha}{\beta_{k-1}}\boldsymbol{\mu}_{k-1}-\boldsymbol{\mu}_{k-1}=\beta_k(\mathbf{x}_k-\boldsymbol{\mu}_{k-1})$. Also note that $\sum_{i=0}^{k-1}\alpha^{k-i}(\mathbf{x}_i-\boldsymbol{\mu}_{k-1})=\frac{\alpha}{\beta_{k-1}}\boldsymbol{\mu}_{k-1}-\boldsymbol{\mu}_{k-1}\alpha\sum_{i=0}^{k-1}\alpha^{k-1-i}=\frac{\alpha}{\beta_{k-1}}\boldsymbol{\mu}_{k-1}-\frac{\alpha}{\beta_{k-1}}\boldsymbol{\mu}_{k-1}=0$.

Using these results, we have:

$$\begin{aligned}\Sigma_k &= \beta_k\sum_{i=0}^{k}\alpha^{k-i}(\mathbf{x}_i-\boldsymbol{\mu}_k)(\mathbf{x}_i-\boldsymbol{\mu}_k)' \\ &= \beta_k\sum_{i=0}^{k}\alpha^{k-i}\begin{pmatrix}(\mathbf{x}_i-\boldsymbol{\mu}_{k-1})(\mathbf{x}_i-\boldsymbol{\mu}_{k-1})'-(\mathbf{x}_i-\boldsymbol{\mu}_{k-1})\mathbf{d}_k' \\ -\mathbf{d}_k(\mathbf{x}_i-\boldsymbol{\mu}_{k-1})'+\mathbf{d}_k\mathbf{d}_k'\end{pmatrix} \\ &= \alpha\frac{\beta_k}{\beta_{k-1}}\Sigma_{k-1}+\beta_k\begin{pmatrix}(\mathbf{x}_k-\boldsymbol{\mu}_{k-1})(\mathbf{x}_k-\boldsymbol{\mu}_{k-1})' \\ -(\mathbf{x}_k-\boldsymbol{\mu}_{k-1})\mathbf{d}_k' \\ -\mathbf{d}_k(\mathbf{x}_k-\boldsymbol{\mu}_{k-1})'\end{pmatrix}+\mathbf{d}_k\mathbf{d}_k' \\ &= \alpha\frac{\beta_k}{\beta_{k-1}}\Sigma_{k-1}+\beta_k(1-\beta_k)(\mathbf{x}_k-\boldsymbol{\mu}_{k-1})(\mathbf{x}_k-\boldsymbol{\mu}_{k-1})'\end{aligned}$$

Then (4) follows using the geometric series summation formula for β_k. ■

Proof of Lemma 3.2.

Define the sum of weights for the fixed window length exponential weighting scheme of length n and decay factor α as $S=\sum_{i=k-n+1}^{k}\alpha^{k-i}$. Note that this is independent of k and can be expressed as $S=\frac{\alpha^n-1}{\alpha-1}$.

We have:

$$\begin{aligned}\Sigma_K &= \frac{1}{S}\sum_{i=k-n+1}^{k}\alpha^{k-i}(\mathbf{x}_i-\boldsymbol{\mu}_k)(\mathbf{x}_i-\boldsymbol{\mu}_k)' \\ &= \frac{\alpha}{S}\sum_{i=k-n}^{k-1}\alpha^{k-1-i}(\mathbf{x}_i-\boldsymbol{\mu}_k)(\mathbf{x}_i-\boldsymbol{\mu}_k)' \\ &\quad +\frac{1}{S}(\mathbf{x}_k-\boldsymbol{\mu}_k)(\mathbf{x}_k-\boldsymbol{\mu}_k)' \\ &\quad -\frac{\alpha^n}{S}(\mathbf{x}_{k-n}-\boldsymbol{\mu}_k)(\mathbf{x}_{k-n}-\boldsymbol{\mu}_k)' \qquad (18)\end{aligned}$$

Now define $\mathbf{d}_k\underline{\underline{\text{def}}}\ \boldsymbol{\mu}_k-\boldsymbol{\mu}_{k-1}$. Then we can express the first term of (18) as:

$$\begin{aligned}&\frac{\alpha}{S}\sum_{i=k-n}^{k-1}\alpha^{k-1-i}(\mathbf{x}_i-\boldsymbol{\mu}_k)(\mathbf{x}_i-\boldsymbol{\mu}_k)' \\ &= \frac{\alpha}{S}\sum_{i=k-n}^{k-1}\alpha^{k-1-i}(\mathbf{x}_i-\boldsymbol{\mu}_{k-1}-d_k)(\mathbf{x}_i-\boldsymbol{\mu}_{k-1}-\mathbf{d}_k)' \\ &= \frac{\alpha}{S}\sum_{i=k-n}^{k-1}\alpha^{k-1-i}\begin{pmatrix}(\mathbf{x}_i-\boldsymbol{\mu}_{k-1})(\mathbf{x}_i-\boldsymbol{\mu}_{k-1})' \\ -(\mathbf{x}_i-\boldsymbol{\mu}_{k-1})\mathbf{d}_k' \\ -\mathbf{d}_k(\mathbf{x}_i-\boldsymbol{\mu}_{k-1})+\mathbf{d}_k\mathbf{d}_k'\end{pmatrix} \\ &= \alpha\Sigma_{k-1}+\alpha\mathbf{d}_k\mathbf{d}_k' \qquad (19)\end{aligned}$$

Where the last line follows since:
$\boldsymbol{\mu}_{k-1}=\frac{1}{S}\sum_{k-n}^{k-1}\alpha^{k-1-i}\mathbf{x}_i$.

So
$$\sum_{k-n}^{k-1} \alpha^{k-1-i}(\mathbf{x}_i - \boldsymbol{\mu}_{k-1}) = S\boldsymbol{\mu}_{k-1} - S\boldsymbol{\mu}_{k-1} = 0.$$

We can also express $\mathbf{d}_k$ in the following manner:

$$\begin{aligned}
\mathbf{d}_k &= \boldsymbol{\mu}_k - \boldsymbol{\mu}_{k-1} \\
&= \boldsymbol{\mu}_k - \frac{1}{S}\sum_{k-n}^{k-1} \alpha^{k-1-i}\mathbf{x}_i \\
&= \boldsymbol{\mu}_k - \frac{1}{S\alpha}\sum_{k-n}^{k-1} \alpha^{k-i}\mathbf{x}_i \\
&= \boldsymbol{\mu}_k - \frac{1}{S\alpha}\left[\sum_{k-n+1}^{k} \alpha^{k-i}\mathbf{x}_i + \alpha^n\mathbf{x}_{k-n} - \mathbf{x}_k\right] \\
&= \boldsymbol{\mu}_k(1 - \frac{1}{\alpha}) + \frac{1}{S\alpha}[\mathbf{x}_k - \alpha^n\mathbf{x}_{k-n}] \qquad (20)
\end{aligned}$$

Equation (5) then follows from (19, 20) and the expression of S via the geometric sum formula for S.

In the case where α^n is small, note that $S \approx 1/(1-\alpha)$. We can use this and further approximate (20) in the following manner:

$$\begin{aligned}
\mathbf{d}_k &= \boldsymbol{\mu}_k(1 - \frac{1}{\alpha}) + \mathbf{x}_k(\frac{1}{\alpha} - 1) \\
&= (\frac{1}{\alpha} - 1)[\mathbf{x}_k - \boldsymbol{\mu}_k] \qquad (21)
\end{aligned}$$

When applied to (5), this then leads to (6). ■

Proof of Lemma 3.3.

Define $\mathbf{v} \underline{\underline{\text{def}}} (\mathbf{x}_k - \boldsymbol{\mu}_k)$, $a \underline{\underline{\text{def}}} \left(\frac{\alpha^{k+1}-\alpha}{\alpha^{k+1}-1}\right)$, and $b \underline{\underline{\text{def}}} \left(\frac{\alpha-1}{\alpha^{k+1}-\alpha}\right)$.

Substituting into Equation (3) from Lemma 3.1. we have:

$$\Sigma_k = a\Sigma_{k-1} + b\mathbf{v}\mathbf{v}' \qquad (22)$$

From Henderson and Searle (1981), we have the Woodbury Matrix Identity (Matrix Inversion Lemma) which can be expressed as

$$(A + b\mathbf{u}\mathbf{v}')^{-1} = A^{-1} - \frac{b}{1 + b\mathbf{v}'A^{-1}\mathbf{u}}A^{-1}\mathbf{u}\mathbf{v}'A^{-1} \qquad (23)$$

Then substituting the values of (22) into (23) and recalling that $(aA)^{-1} = a^{-1}A^{-1}$ for any nonsingular matrix (A) and non-zero scalar a, gives the result of (7). ■

Proof of Lemma 3.4.

The proof again uses the Woodbury Matrix Identity from (23), and follows similar steps to the proof of Lemma 3.3. It is not listed here for brevity. ■

Proof of Lemma 3.5.

Note the block matrix inversion formula from Henderson and Searle (1981)

$$\begin{bmatrix} A & U \\ V & D \end{bmatrix}^{-1} = \begin{bmatrix} (A - UD^{-1}V)^{-1} & -A^{-1}U(D - VA^{-1}U)^{-1} \\ -D^{-1}V(A - UD^{-1}V)^{-1} & (D - VA^{-1}U)^{-1} \end{bmatrix} \qquad (24)$$

Substituting the block description of Σ_A from (9) into (24) gives:

$$\Sigma_A^{-1} = \begin{bmatrix} \Sigma_O & \Sigma_{ON} \\ \Sigma'_{ON} & \Sigma_N \end{bmatrix}^{-1} = \begin{bmatrix} \left(\Sigma_O - \Sigma_{ON}\Sigma_N^{-1}\Sigma'_{ON}\right)^{-1} & -\Sigma_O^{-1}\Sigma_{ON}\left(\Sigma_N - \Sigma'_{ON}\Sigma_O^{-1}\Sigma_{ON}\right)^{-1} \\ -\Sigma_N^{-1}\Sigma'_{ON}\left(\Sigma_O - \Sigma_{ON}\Sigma_N^{-1}\Sigma'_{ON}\right)^{-1} & \left(\Sigma_N - \Sigma'_{ON}\Sigma_O^{-1}\Sigma_{ON}\right)^{-1} \end{bmatrix}$$

The above equation would still require the inverse of a (large) n by n matrix for the left hand blocks. This large inverse is avoided by using the Matrix Inversion Lemma, as shown in Henderson and Searle (1981):

$$\begin{aligned}
&\left(A - UD^{-1}V\right)^{-1} \\
&= A^{-1} + A^{-1}U\left(D - VA^{-1}U\right)^{-1}VA^{-1} \qquad (25)
\end{aligned}$$

Applying (25) to the matrix inverse required for the left hand blocks of (25) provides Equation (10). ■

Proof of Lemma 3.6.

Following the proof of Lemma 3.5. we established the formulae for Θ and Λ. Also from Equation (10) we can see that $\Phi = -\Sigma_O^{-1}\Sigma_{ON}\Theta$.

First we show that Θ is symmetric. Note that the inverse of a symmetric matrix is symmetric. $\Theta^{-1} = \Sigma_N - \Sigma'_{ON}\Sigma_O^{-1}\Sigma_{ON}$. We know that Σ_N and Σ_O are symmetric since they are covariance matrices. The symmetry of the equation then follows by inspection.

Then from the definitions for those variables, we have

$$\begin{aligned}
&\Lambda - \Phi\Theta^{-1}\Phi' \\
&= \Sigma_O^{-1} + \Sigma_O^{-1}\Sigma_{ON}\Theta\Sigma'_{ON}\Sigma_O^{-1} \\
&\quad - \Sigma_O^{-1}\Sigma_{ON}\Theta\Theta^{-1}\Theta'\Sigma'_{ON}\Sigma_O^{-1} \\
&= \Sigma_O^{-1} + \Sigma_O^{-1}\Sigma_{ON}\Theta\Sigma'_{ON}\Sigma_O^{-1} \\
&\quad - \Sigma_O^{-1}\Sigma_{ON}\Theta\Sigma'_{ON}\Sigma_O^{-1} \\
&= \Sigma_O^{-1} \qquad (26)
\end{aligned}$$

Where the result follows since Θ is symmetric. ■

Proof of Lemma 3.7.

The proof follows from the definition of covariance, and the fact that for a permutation matrix, $P^{-1} = P'$. ■