Practical Portfolio Optimization

K V Fernando

NAG Ltd
Wilkinson House
Jordan Hill
Oxford OX2 8DR
United Kingdom

email: vince@nag.co.uk

Abstract

NAG Libraries have many powerful and reliable optimizers which can be used to solve large portfolio optimization and selection problems in the financial industry. These versatile routines are also suitable for academic research and teaching.

Key words
Markowitz, mean-variance analysis, optimal portfolios, minimum variance portfolio, portfolio selection, portfolio allocation, portfolio diversification, portfolio optimization, efficient frontier, mean-variance frontier, MV efficiency

Contents

1 Introduction

The selection of assets or equities is not just a problem of finding attractive investments. Designing the correct portfolio of assets cannot be done by human intuition alone and requires modern, powerful and reliable mathematical programs called optimizers. The Numerical Algorithms Group Ltd (NAG) is world renowned for its work on numerical algorithms, and NAG routines for optimization are being used extensively in industry, commerce and academia. Many leading financial companies and institutions employ NAG optimizers to select, diversify and rebalance their portfolios. They are also used by business and management schools for teaching and research.

Any investor would like to have the highest return possible from an investment. However, this has to be counterbalanced by the amount of risk the investor is able or desires to take. The expected return and the risk measured by the variance (or the standard deviation, which is the square-root of the variance) are the two main characteristics of a portfolio. Unfortunately, equities with high returns usually correlate with high risk.

The behaviour of a portfolio can be quite different from the behaviour of individual components of the portfolio. The risk of a properly constructed portfolio from equities in leading markets could be half the sum of the risks of individual assets in the portfolio. This is due to complex correlation patterns between individual assets or equities. A good optimizer can exploit the correlations, the expected returns, the risk (variance) and user constraints to obtain an optimized portfolio. NAG optimization routines can deliver optimized and diversified portfolios to match investor expectations.

The mathematical problem of portfolio optimization was initiated by Professor Harry Markowitz in the fifties and he was rewarded with a Nobel Prize in Economics in 1990 which he shared with Professors William Sharpe and Merton Miller [8]. NAG optimizers can handle the classical Markowitz optimization problems [7], [9], [10] and many modern day extensions [4], [11], [13], [14], [15]. NAG also provides a consultancy service to the financial sector to solve mathematical, numerical, programming and visualization problems associated with portfolio optimization.

Portfolio optimization is often called mean-variance (MV) optimization. The term mean refers to the mean or the expected return of the investment and the variance is the measure of the risk associated with the portfolio. The mathematical problem can be be formulated in many ways but the principal problems can be summarized as follows.

1. Minimize risk for a specified expected return

2. Maximize the expected return for a specified risk

3. Minimize the risk and maximize the expected return using a specified risk aversion factor

4. Minimize the risk regardless of the expected return
5. Maximize the expected return regardless of the risk
6. Minimize the expected return regardless of the risk

The above problems could have linear or nonlinear constraints, equality and inequality constraints.

The first three problems are essentially mathematically equivalent and the solutions are called mean-variance (MV) efficient. The efficient points in the Return-Risk graph are called the *Efficient Frontier*.

The fourth problem gives minimum variance solutions which are for cautious investors. It is also used for comparison and benchmarking of other portfolios. The fifth problem gives the upper bound of the expected return which can be attained; this is also useful for comparisons. The last problem indicates a worst case scenario.

When market conditions or the investors risk preferences change, it is advisable to rebalance the portfolio. Any of the above problems can be solved relative to an existing portfolio or a benchmark.

The transactions costs associated with purchasing a new portfolio or rebalancing a portfolio could represent a significant amount to the investor. NAG optimization routines can handle transactions costs and they could significantly affect the composition of the portfolio.

2 NAG Routines for Optimization

2.1 A Selection of Library Routines

NAG Libraries devote a complete Chapter to function optimization. Table 1 summarizes the routines available in NAG Libraries.

Routine name	Constraints	Objective function	Dense or sparse
e04nc	linear	quadratic	dense
e04nf	linear	quadratic	dense
e04nk	linear	quadratic	sparse
e04uc	nonlinear	nonlinear	dense
e04uf	nonlinear	nonlinear	dense
e04ug	nonlinear	nonlinear	sparse

Table 1: Optimization routines

2.2 Quadratic Programming with Linear Constraints

The e04nf routine is mainly for Linear Programming (LP) and Quadratic Programming (QP), and the following objective functions can be minimized.

LP1: $c^t x$. This option can be used for finding the maximum or the minimum expected return regardless of the variance. The vector c is set to the expected return vector a. The weight vector of the portfolio is represented by x.

QP1: $\frac{1}{2}x^t Hx$. This option can be used to minimize the variance; the matrix H is set to the covariance matrix V.

QP2: $\frac{1}{2}x^t Hx + c^t x$. This option can be used to minimize the variance for a specified risk aversion factor; the matrix H is set to the covariance matrix V and c is set to $-\lambda a^t x$ where λ is the risk aversion parameter and a is the expected return vector. Sometimes, the objective function is defined as $\frac{1}{2}(x^t Vx - \mu a^t x)$; in that case, the new risk aversion factor μ is equal to 2λ.

QP3: $\frac{1}{2}x^t H^t Hx$. This is similar to QP1 except that the variance is supplied via the Cholesky factor H of the covariance matrix V. See Section 5.1 for more details.

QP4: $\frac{1}{2}x^t H^t Hx + c^t x$. This is similar to QP2 except that the covariance is supplied via the Cholesky factor H of the covariance matrix V.

The routine e04nc can also handle least squares minimization or regression problems. Many portfolio optimization problems can be formulated as least squares or regression problems. Hence the same routine e04nc can be used for solving general portfolio optimization as well as portfolio problems which are posed as least squares or regression problems.

2.3 Nonlinear Programming

The more general routines e04uc and e04uf can accommodate nonlinear objective functions with nonlinear constraints.

2.4 Routines for Sparse Matrix Problems

Most of the optimization routines in NAG Libraries are for dense matrix problems; that is, the constraint matrix is assumed to have mostly non-zero elements. However, the routines e04nk and e04ug are designed to work with sparse constraint matrices. The routine e04nk is for linear and quadratic programming and the routine e04ug is for nonlinear programming. The routine e04nk does not explicitly require the covariance matrix V but the user has to supply the matrix-vector product Vx.

2.5 Forward and Reverse Communication

Most of the optimization routines are based on forward communications. In such programs, the routine is called only once to obtain the results and the user supplies all the necessary information to the NAG routine via a subroutine.

However, in some circumstances, it is necessary to do the optimization step by step and call the user routine repeatedly to get fresh information. The NAG routine e04uc is a forward communication routine and e04uf is the reverse communication equivalent. This reverse communication routine is particularly useful when it is called from another language (i.e., Microsoft VBA) which does not fully support procedure arguments in a way that is compatible with NAG routines.

2.6 Hardware, Operating Systems and Environments

NAG Libraries are available for all popular hardware systems and operating systems. There are implementations for PCs, workstations, parallel computers such as SGI/Cray T3E and IBM SP3, networks of workstations, and powerful SMP machines. Various operating systems from Windows (2000, 98, NT), DOS and Linux for PCs to UNIX and VMS for larger systems are supported.

All of the optimization routines are available as DLLs.

3 Interfaces to Routines

3.1 Portfolio Weights

The weight or the proportion by value of the holding of the asset i is denoted by x_i.

3.2 Primary Data

NAG optimization routines require the following details about the portfolio.

n number of assets in the portfolio

a expected returns of the assets in a vector of length n

s volatilities (standard deviations) of the assets in a vector of length n

C correlation coefficients in an n by n symmetric matrix

l_i lower limit of the weight of asset i

u_i upper limit of the weight of asset i

However, instead of the volatility vector s and the correlation matrix C, the following may be available:

V covariance matrix in an n by n symmetric matrix

Usually, NAG optimization routines require the covariance matrix V, and if it is not available then it can be easily computed from the standard deviation vector s and the correlation matrix C. However, in Section 5.1, we advocate that the covariance matrix V should not be formed explicitly to avoid information loss; what is required is the Cholesky factor R of the covariance matrix V and not the matrix V itself.

3.3 General Linear Constraints

The primary equality constraint is that all the weights should add up to a constant, usually to unity. That is

$$\sum_{i=1}^{n} x_i = 1$$

and the above is known as the budget constraint or the investment constraint. In matrix notation

$$e^t x = 1$$

where the column vector x is the weight vector which holds the proportions of the assets

$$x = (x_1, x_2, \ldots, x_n)^t$$

and e is a column vector with all elements equal to unity.

$$e = (1, 1, \ldots, 1)^t.$$

The expected return vector is denoted by

$$a = (a_1, a_2, \ldots, a_n)^t$$

where a_i contains the the expected return of the asset or the equity i.

NAG optimizers can handle many equality constraints, and inequality constraints and they are of the form

$$\begin{array}{rcl} l \leq & x & \leq u \\ L \leq & Ax & \leq U \\ \text{where} \quad l & = & (l_1, \ldots, l_n)^t \\ u & = & (u_1, \ldots, u_n)^t \\ L & = & (L_1, \ldots, L_m)^t \\ U & = & (U_1, \ldots, U_m)^t \end{array}$$

l_i and L_i are lower limits, u_i and U_i represent the upper limits, and m denotes the number rows of the matrix A. For equality constraints, lower and upper limits are set to equal values. It is also possible to set the upper limits to $+\infty$ and the lower limits to $-\infty$

These constraints could represent sector, industry, country, exposure (momentum, QRQ, P/B) and other user-defined financial constraints.

3.4 Nonlinear Constraints

NAG optimization routines, in particular routines e04uc, e04uf and e04ug, can handle nonlinear constraints; they also can support nonlinear objective functions.

3.5 Cold and Warm Starts

Cold starts refer to solutions of the problem from scratch. However, if the routines are called repeatedly then approximate solutions are available from previous solutions. In that case, the initial conditions for the next iteration may be supplied from the previous. Such son-called warm start facilities are available for many NAG optimization routines.

4 The Optimization Problems

Here we give some of the problems which can be solved using NAG optimizers. However, this list is not comprehensive and many more optimization problems can be solved using NAG software. For the mathematically minded, these problems can be stated as follows.

Problem 1 (Minimize the Risk)

$$\textit{minimize the variance } \; x^t V x$$

with a specified expected return $r = a^t x$ *subjected to linear and (or nonlinear) constraints.*

Problem 2 (Maximize the Expected Return)

$$\textit{maximize the expected return } \; r = a^t x$$

with a specified variance $\nu = x^t V x$. *This is also equivalent to minimization of* $-r$.

Problem 3 (Maximize Expected Return with Risk Aversion)

$$\textit{maximize } \; \lambda a^t x - x^t V x$$

subjected to linear and (or nonlinear) constraints where λ *is the risk aversion parameter. This is also equivalent to*

$$\textit{minimize } \; x^t V x - \lambda a^t x.$$

Problem 4 (Minimize Risk)

$$\textit{minimize the variance } x^tVx$$

subjected to linear and (or nonlinear) constraints.

Problem 5 (Maximize the Expected Return)

$$\textit{maximize the expected return } r = a^tx$$

subjected to linear and (or nonlinear) constraints. This is also equivalent to

$$\textit{minimize } \; -a^tx.$$

Problem 6 (Minimize the Expected Return)

$$\textit{minimize the expected return } r = a^tx$$

subjected to linear and (or nonlinear) constraints.

The above six problems can be solved with respect to a benchmark or the existing portfolio. The following is an example and many more variants are possible.

Problem 7 (Minimize Risk with Respect to a Benchmark)

$$\textit{minimize } \; (x - x_b)^tV(x - x_b)$$

with a specified expected relative return r defined by

$$r = a^t(x - x_b)$$

subjected to linear and (or nonlinear) constraints where x_b represents a benchmark portfolio.

5 Processing of Raw Data

5.1 The Covariance Matrix in Factored Form

Let the element $y_{i,j}$ of the matrix Y denotes the return of the asset j at time i. If there are m time periods and n assets then the matrix Y is an m by n matrix. The sample mean for asset j is given by

$$a_j = \frac{1}{m}\sum_{i=1}^{m} y_{i,j}.$$

Once the sample mean is removed from the returns, we get the deviation from the sample mean. This deviation $\hat{Y}$ can be obtained as

$$\hat{Y} = Y - ea^t$$

where e is a column vector of length m with each and every element equal to unity,

$$e = (1, \ldots, 1)^t.$$

The sample covariance matrix V is usually computed as

$$V = \frac{1}{m}\hat{Y}^t\hat{Y}$$

but some statisticians might prefer the formula

$$V = \frac{1}{m-1}\hat{Y}^t\hat{Y}.$$

However, it is well known to numerical analysts that such computation of covariance matrices lead to loss of information; see [1], [5] or [6].

Here is a well known example where information is lost in the formation of the covariance. Let $\hat{Y}$ be

$$\hat{Y} = \begin{pmatrix} 1 & 1 & 1 \\ \theta & & \\ & \theta & \\ & & \theta \end{pmatrix}$$

which is full rank provided that θ is non-zero. The covariance matrix is then given by

$$\hat{Y}^t\hat{Y} = \frac{1}{3}\begin{pmatrix} 1+\theta^2 & 1 & 1 \\ 1 & 1+\theta^2 & 1 \\ 1 & 1 & 1+\theta^2 \end{pmatrix}.$$

In IEEE machines with double precision floating point arithmetic the machine precision ϵ is approximately 2^{-53}. If θ is smaller than $\sqrt{\epsilon}$, then in floating point arithmetic, the computer will determine the diagonal elements of $\hat{Y}^t\hat{Y}$ as

$$1 + \theta^2 \longrightarrow 1$$

due to rounding errors. In that case, the covariance matrix is evaluated as

$$\hat{Y}^t\hat{Y} = \frac{1}{3}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \frac{1}{3}ee^t$$

which is a rank-one matrix and hence singular. Thus the explicit formulation of the covariance matrix should be avoided. One of the ways to avoid such information loss is to compute the QR factorization of $\hat{Y}$, which is defined by

$$QR = \hat{Y}$$

where the m by n matrix Q contains orthogonal columns (i.e., $Q^tQ = I$) and R is an n by n upper triangular matrix. There are routines in NAG Libraries (e.g., f08ae or f01qc) to compute the QR factorization. The covariance matrix is then given by

$$V = \frac{1}{m} R^t Q^t Q R = \frac{1}{m} R^t R.$$

However, the above product does not have to be computed explicitly in portfolio optimization, and the Cholesky factor $R/\sqrt{m}$ can be given as an input to optimization routines. In this case, the optimization option QP3 or QP4 (see Section 2.2) should be used. Note that it is not necessary to compute the matrix Q explicitly.

5.2 Determination of the Singular Values of the Cholesky Factors

To check whether the covariance matrix is positive definite, the singular values of the Cholesky factor R may be computed. The eigenvalues of the covariance matrix H are the squares of the singular values of the Cholesky factors. Thus the eigenvalues of the covariance matrix may be determined without explicitly forming the covariance matrix.

5.3 If the Covariance Matrix Already Exists

We have already indicated in a previous section that the covariance matrix should not be formed explicitly. However, if it has been already formed then it can be given as an input to NAG optimization routines directly using the options QP1 and QP2 (see Section 2.2 for further details). However, if NAG optimizations are to be called more than once then it is more efficient to use QP3 and QP4 options. In that case, the Cholesky factors of the covariance matrix

$$V = R^t R$$

should be determined prior to calling the optimization routines. This can be done using f07fd or f03ae if the covariance matrix is positive definite. If the factorization does not exist then these routines will indicate the failure via an appropriate error handling flag.

Covariance matrices are usually positive definite or at least positive semi-definite. However, due to rounding errors covariance matrices might not be positive definite, and in that case the the factorization routines might fail to compute the factors. However, there is no guarantee that the covariance matrix is positive definite even if the Cholesky factors exist. Consider the example

$$H = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} = R^t R.$$

The covariance matrix H has a Cholesky factor R but it is not positive definite.

5.4 Eigenvalues of the Covariance Matrix

We have already indicated that to avoid information loss the covariance matrix should not be formed explicitly. However, if it has been already formed then the best way to find whether the covariance matrix is positive definite or not is to compute the eigenvalues of the covariance matrix. If the eigenvalues are positive then the matrix is positive definite. However, even if all eigenvalues are positive, it is advisable to compute the condition number of the matrix (which is defined as the largest eigenvalue divided by the smallest eigenvalue). If the condition number is high then the matrix is nearly positive semi-definite and this could lead to results which are unreliable.

The eigenvalues may be computed using f02aa but there are also many other routines that could be used.

5.5 Missing Values

In Section 5.1, a matrix Y was defined, and it contains the data to compute the covariance matrix and the Cholesky factors of the covariance matrix. In practice, this matrix may contain missing (unrecorded) data. If this is the case then a suitable data interpolation routine has to be used to assign some values to the missing entries. More sophisticated users might find the missing values using the Brownian bridge EM algorithm [12] which is an adaptation of the classical EM algorithm [3]) for Brownian processes in finance.

6 Numerical Examples: Selection of Equities

For illustrative purposes, we give a few examples of the results which could be expected from NAG optimizers. The data was taken from [2] where portfolios of shares in the Hang Seng (Hong Kong, 31 shares), DAX 100 (Germany, 85 shares), FTSE 100 (UK, 89 shares), S & P 100 (USA, 98 shares) and the Nikkei (Japan, 225 shares) were studied using NAG e04 routines. Our *efficient frontiers* are based on the routine e04nf.

The data refers to weekly returns and weekly variance. Figures 1 to 5 give the efficient frontiers for this data. The dominant (red) curve denotes the frontier if the weights are constrained between zero and unity (no short sales are allowed). When tighter constraints are forced on the weights ($0.0 \leq x_i \leq 0.1$), a different frontier appears. In general, with more constraints, the expected returns become smaller for a specified risk value. Individual points denote individual shares; the risk reduction due to optimization can be judged by comparing these shares with the efficient frontier in these figures.

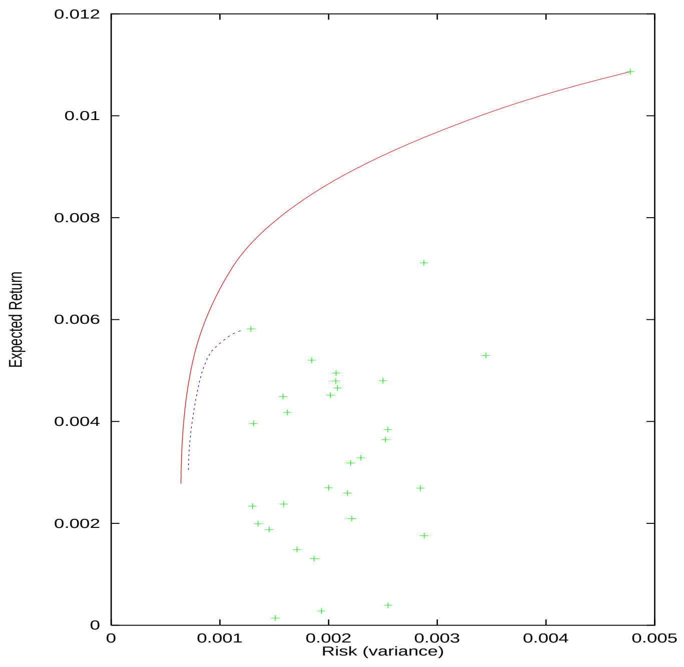

Figure 1: The Efficient Frontiers for the Hang Seng example

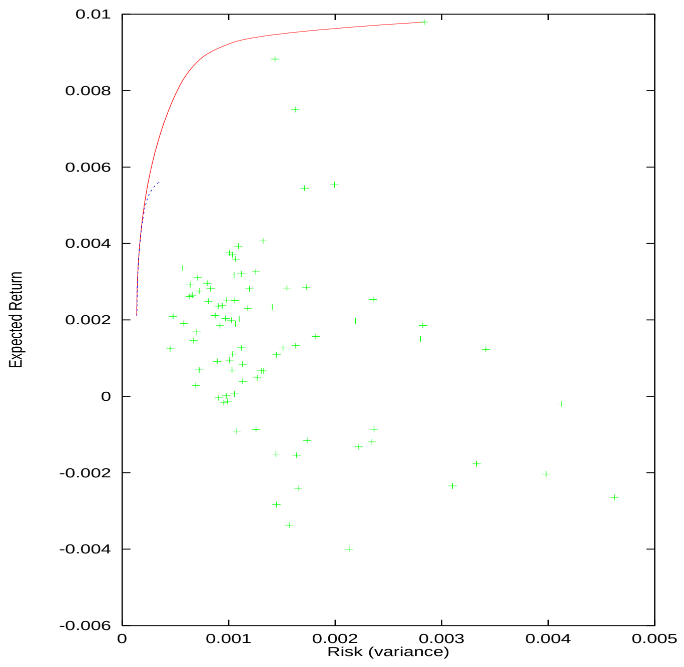

Figure 2: The Efficient Frontiers for the DAX 100 example

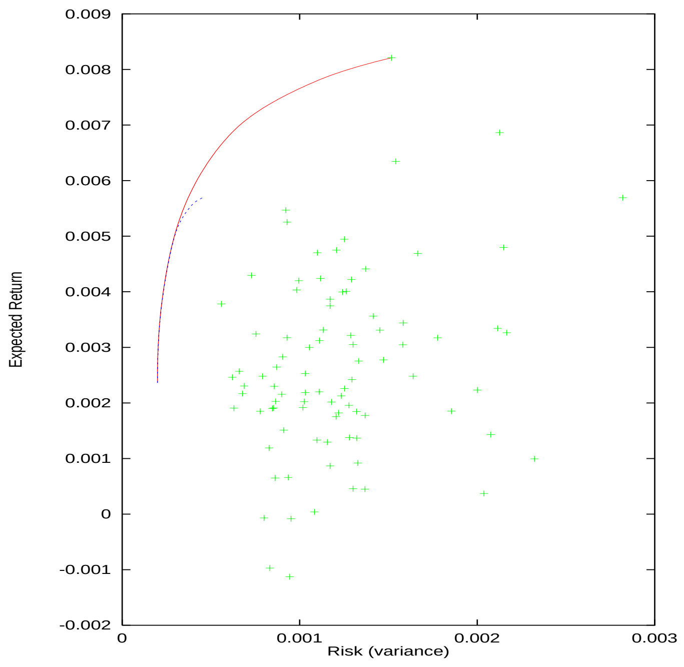

Figure 3: The Efficient Frontiers for the FTSE 100 example

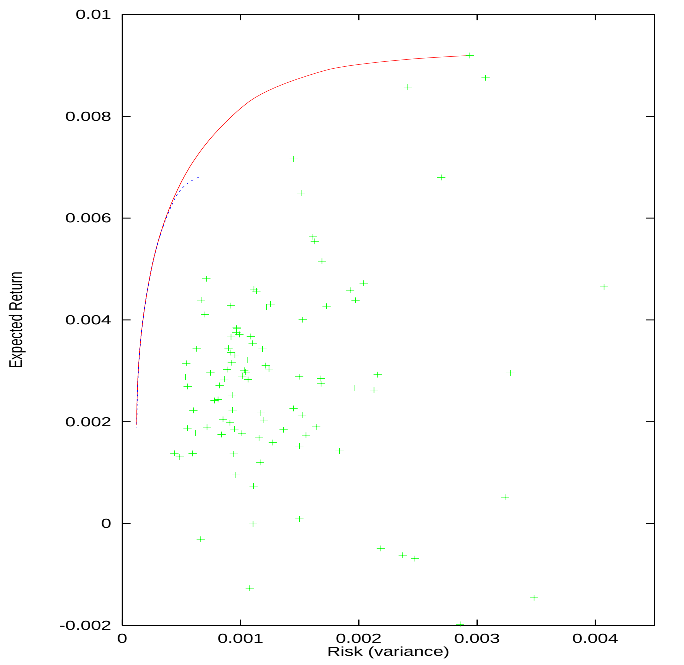

Figure 4: The Efficient Frontiers for the S & P 100 example

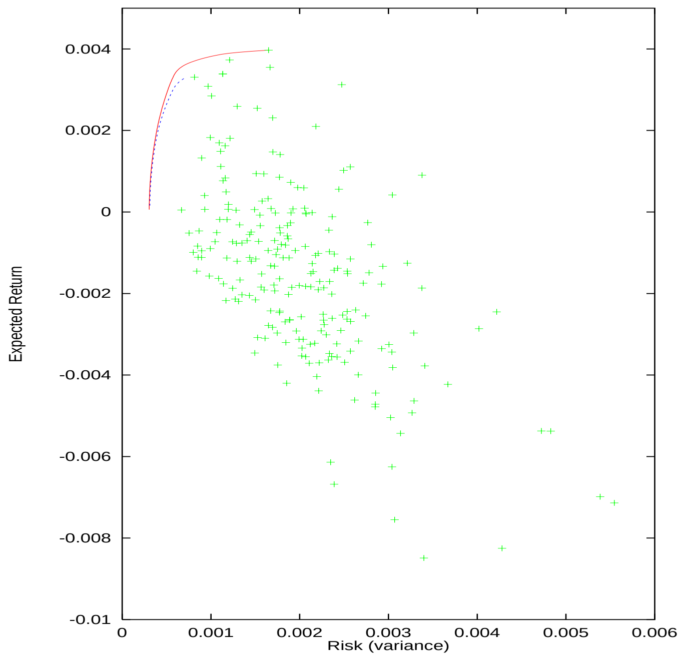

Figure 5: The Efficient Frontiers for the Nikkei example

In practical problems, with more financial constraints, the efficient frontier reduces to a smaller curve or almost to a point.

For the Nikkei example, we have also computed the Inefficient Frontier, which is obtained by using negative values for the risk aversion parameter λ; see Problem 3 in Section 4. In Figure 6, the top half of the curve (in red) shows the Efficient Frontier and the bottom part of the curve (in blue) gives the Inefficient Frontier. No short sales were allowed in this example.

Markowitz-type optimization is frequently used for asset allocation. The asset manager could have equities, government and corporate bonds, international equities and bonds, real estate, venture capital and gold to select the portfolio of assets. Each asset could itself be a portfolio (with or without optimization).

For our next example, we have used eight types of assets (US 3 month treasury bills, US government long bonds, SP 500, Wilshire 500, Nasdeq composite corporate bond index, EAFE and Gold). The yearly returns are from 1973 to 1994. These could be found at
http://www.sor.princeton.edu/ rvdb/ampl/nlmodels/markowitz/

Figure 7 shows two efficient frontiers. The dominant (red) curve shows the frontier with upper limits of assets set to unity and the lower limits to zero. When the lower limit is raised to 0.05 and the upper limit is lowered to 0.5, we get the efficient frontier shown with a dotted curve (in blue).

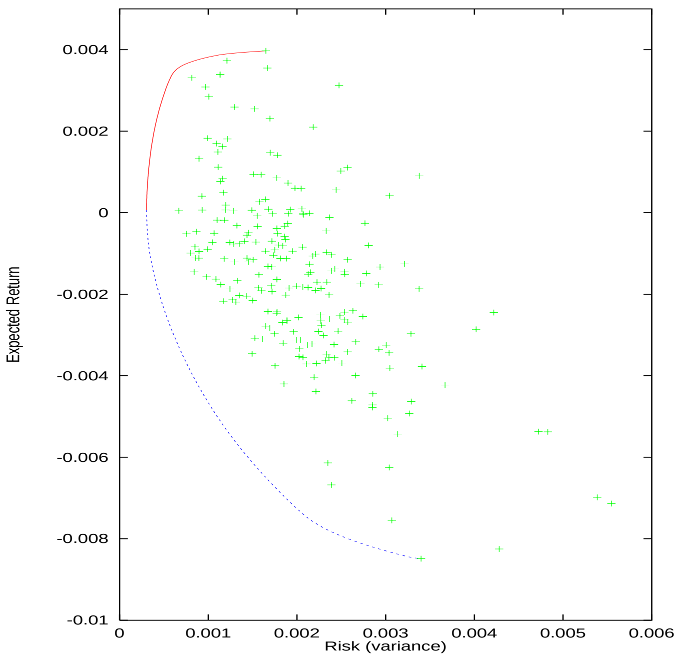

Figure 6: The Efficient and Inefficient Frontiers for the Nikkei example

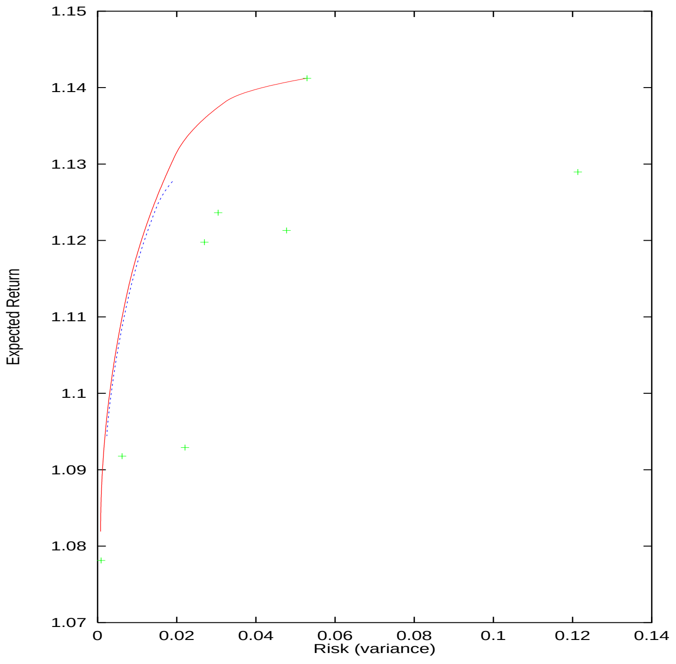

Figure 7: The Efficient Frontiers in asset allocation

7 Numerical Example: Asset Allocation

In this numerical example, we started with raw data (yearly returns) and did not form the covariance matrix explicitly. Instead, as described in Section 5.1, only the Cholesky factor R of the covariance was computed. In this case, the optimization option QP3 or QP4 (see Section 2.2) should be used since only the Cholesky factor R and not the whole covariance matrix V has to be given as an input. Figure 7 exhibits the efficient frontier for this problem.

8 Transactions Costs

In the classical work of Markowitz, transactions costs associated with buying and selling of equities were not allowed. However, now the importance of incorporating transactions costs in building new portfolios and also in rebalancing existing portfolios are well recognized. In general, transactions costs are not trivial enough to be neglected, and the optimal portfolio depends upon the total cost of transactions.

Let the transactions cost associated with buying the equity i be p_i and then it may be modelled as

$$p_i = \begin{cases} (x_i - \bar{x}_i)g_i & \text{for} \quad x_i > \bar{x}_i \\ 0 & \text{for} \quad x_i \le \bar{x}_i \end{cases}$$

where x_i is the new portfolio weight of the equity i, $\bar{x}_i$ is the original weight of the equity i, and g_i is a constant associated with buying the equity i. Similarly, let q_i be the cost of selling the equity i,

$$q_i = \begin{cases} (\bar{x}_i - x_i)i_i & \text{for} \quad x_i < \bar{x}_i \\ 0 & \text{for} \quad x_i \ge \bar{x}_i \end{cases}$$

where the constant h_i is associated with selling. Note that both p_i and q_i cannot be simultaneously non-zero since the buying cost is zero when there is selling and vice versa.

Let $\phi(x)$ be the objective function for minimization without transaction costs. The new objective function with transaction costs is then given by

$$\phi(x) + \sum_{i=1}^{n} p_i + q_i = \phi(x) + \sum_{i=1}^{n} \max\{p_i, q_i\}.$$

The above objective function is not smooth (non-differentiable). However, this can be transformed into a smooth problem by including a new variable y_i for each equity i. The new objective is then given by

$$\phi(x) + \sum_{i=1}^{n} y_i$$

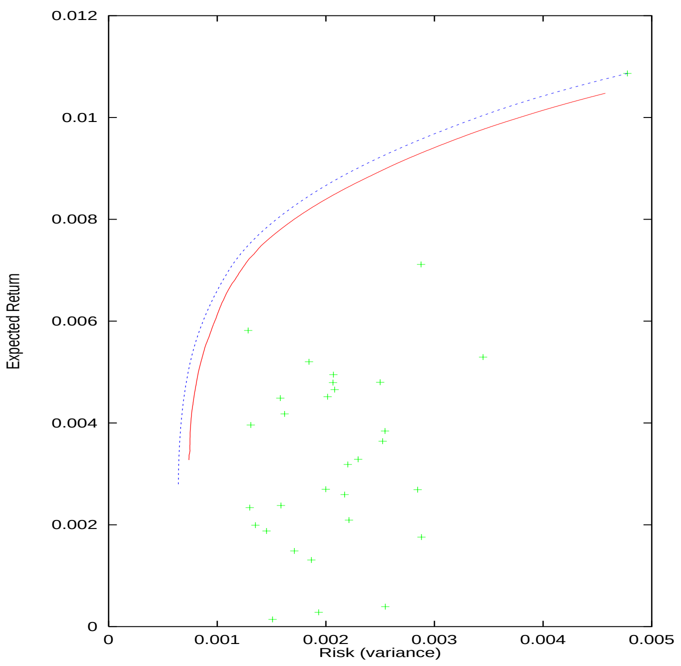

Figure 8: The Efficient Frontiers with and without Transactions Costs

subjected to the constraints

$$p_i \leq y_i$$

and

$$q_i \leq y_i$$

for $i = 1, \ldots, n$, where y_i is the transaction cost for equity i. The above two inequalities can be written in the format

$$\begin{array}{rcl} -\infty \leq & g_i x_i - y_i & \leq g_i \bar{x}_i \\ h_i \bar{x}_i \leq & y_i + h_i x_i & \leq +\infty \end{array}$$

Unfortunately, this approach doubles the number of variables for optimization.

We have recomputed the efficient frontier for the Hang Seng example by including transactions costs. We have assumed that p_i (which determines the buying costs) is equal to 0.0001 and q_i (which determines the selling costs) is equal to 0.0002 for each equity i. Furthermore, it was assumed that the initial portfolio had an equal weighting, $x_i = 1/38$, for each equity i. Figure 8 shows the efficient frontiers with and without transactions costs. For a given variance, the difference between the expected returns (for a given variance) reflects the transactions costs.

9 An Example Program

It is fairly easy to use NAG routines to do practical portfolio optimization. The following example routine shows how to compute the optimal portfolio when the expected return is minimized (i.e., the problem 1 in Section 1). This program may be run in conduction with data used in [2]). In addition, the user has to specify the expected return of the portfolio. The program outputs the optimal weights and the variance of the portfolio.

```
/* Example Program for Portfolio Optimization
 *
 * Copyright 2000 Numerical Algorithms Group Ltd.
 *
 * This example program computes the optimal weights
 * of the equities for a specified expected return
 * of the portfolio */

#include <Nag/nag.h>
#include <stdio.h>
#include <Nag/nag_stdlib.h>
#include <Nag/nag_string.h>
#include <Nag/nage04.h>

static void ex1(void);
static void ex1();

#define MAXN 50
#define MAXLIN 2
#define MAXBND MAXN+MAXLIN

main()
{
  Vprintf("Portfolio optimization using e04nfc.\n");
  ex1();
  exit(EXIT_SUCCESS);
}

static void ex1()
{
  double x[MAXN], cvec[MAXN], s[MAXN];
  double a[MAXLIN][MAXN], h[MAXN][MAXN];
  double bl[MAXBND], bu[MAXBND];
  double objf,up,low,ret,tem,tex,hh;
  Integer tda, tdh;
  Integer i, j, n, nclin, nbnd, ii, jj;
  Boolean print;
  Nag_E04_Opt options;
  static NagError fail, fail2;

  Vscanf(" %*[^\n]"); /* Skip heading in data file */
  Vscanf(" %*[^\n]"); /* Skip heading in data file */

  fail.print = TRUE;
  fail2.print = TRUE;

  /* Read the actual problem dimension.
   * n = the number of equities.  */
  Vscanf("%ld",&n);

  /* nclin = the number of general linear constraints .  */

  nclin =  2;
  nbnd = n + nclin;
  tda = MAXN;
  tdh = MAXN;

  /* a      = the linear constraint matrix A.
   * bl     = the lower bounds on x and A*x.
   * bu     = the upper bounds on x and A*x.
   * x      = the initial estimate of the solution.
   * s      = the standard deviation of return

     For simplicity we assume that
       lower bound on  x is low
       upper bound for x is up
       for all x
     Read low and up */

   Vscanf(" %*[^\n]"); /* Skip heading in data file */
   Vscanf("%lf%lf",&low,&up);

  /*  Set the initial weights to 1/n
      set bl and bu
      set the second row of A to unity
      set the linear constraints bl[n+1] and bu[n+1] */

  for (i = 0; i < n; ++i) {
      x[i] = 1.0/n;
      bl[i] = low;
      bu[i] = up;
      a[1][i] = 1.0;
  }
      bl[n+1] = 1.0;
      bu[n+1] = 1.0;

  /* Read the the expected return and sd
     The expected return is in first column of A */

  Vscanf(" %*[^\n]"); /* Skip heading in data file */
  for (i = 0; i < n; ++i) {
      Vscanf("%lf%lf",&a[0][i],&s[i]);
  }

  /* Read the correlations
     and form the covariance matrix */

  Vscanf(" %*[^\n]"); /* Skip heading in data file */
  for (i = 0; i < n; ++i)
      for (j = i; j < n; ++j) {
         Vscanf("%ld%ld%lf",&ii,&jj,&hh);
         h[ii-1][jj-1]  = hh*s[i]*s[j];
         h[jj-1][ii-1]  = hh*s[i]*s[j];
  }

  /* Read the required expected return of the portfolio
     and set that in bl[n+1] and bu[n+1] */

   Vscanf(" %*[^\n]"); /* Skip heading in data file */
   Vscanf("%lf",&ret);
   bl[n] = ret;
   bu[n] = ret;

  e04xxc(&options); /* Initialise options structure */

  /* Set one option directly
   * Bounds  >=   inf_bound will be treated as plus  infinity.
   * Bounds  <=  -inf_bound will be treated as minus infinity.  */

  options.inf_bound = 1.0e21;

  /* Obtain remaining settings */
  fail.print = TRUE;
```

```
  print = TRUE;
  e04xyc("e04nfc", "stdin", &options, print, "stdout", &fail);

  /* Set the problem type  and other optional settings*/

  options.prob = Nag_QP1;

  /* Solve the problem from a cold start.  */

  if (fail.code == NE_NOERROR)
    e04nfc(n, nclin, (double *)a, tda, bl, bu, cvec, (double *)h, tdh,
           NULLFN, x, &objf, &options, NAGCOMM_NULL, &fail);

    Vprintf("Expected Return   = %lf  \n",ret);
    Vprintf("Expected Variance = %lf  \n",2.0*objf);
  /* Free memory allocated by e04nfc to pointers in options */
  e04xzc(&options, "all", &fail2);

  if (fail.code != NE_NOERROR || fail2.code != NE_NOERROR) exit(EXIT_FAILURE);
} /* ex1 */
```

.

10 Acknowledgements

Discussions with Anne Trefethen, David Sayers, George Levy and Neil Swindells are gratefully acknowledged.

References

[1] A. Björk. *Numerical Methods for Least Squares.* SIAM, Philadelphia, 1996.

[2] T. J. Chang, N. Meade, J. E. Beasley, and Y. M. Sharaiha. Heuristics for cardinality constrained portfolio optimization. *Computers and Operations Research*, 2000, to appear.

[3] A. P. Dempster, N. M. Laired, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B (Methodological)*, 39:1–38, 1977.

[4] E. J. Elton and M. J. Gruber. *Modern Portfolio Theory and Investment Analysis.* Wiley, New York, 1995.

[5] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, 1996.

[6] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problem.* SIAM, Philadelphia, 1995.

[7] H. Markowitz. Portfolio selection. *Journal of Finance*, 7:77–91, 1952.

[8] H. Markowitz, W. F. Sharpe, and M. Miller. *Founders of Modern Finance: Their Prize Winning Concepts and 1990 Nobel Lectures.* AIMR, Charlottesville VA, 1991.

[9] H. M. Markowitz. *Mean-Variance Analysis in Portfolio Choice and Capital Markets.* Blackwell, Oxford, 1987.

[10] H. M. Markowitz. *Portfolio Selection: Efficient Diversification of Investments.* Blackwell, Oxford, 1991.

[11] R. O. Michaud. *Efficient Asset Management.* Harvard Business School Press, Boston, 1998.

[12] W. Morokoff. The Brownian bridge e-m algorithm for covaraince estimation with missing data. *Journal of Computational Finance*, 2:75–100, 1998.

[13] W. F. Sharpe. *Portfolio Theory and Capital Markets.* McGraw-Hill, New York, 1999.

[14] W. F. Sharpe, G. J. Alexander, and J. V. Bailey. *Investments.* Prentice Hall, Upper Saddle River, NJ, 1998.

[15] S. A. Zenios. *Financial Optimization.* Cambridge University Press, Cambridge, 1993.