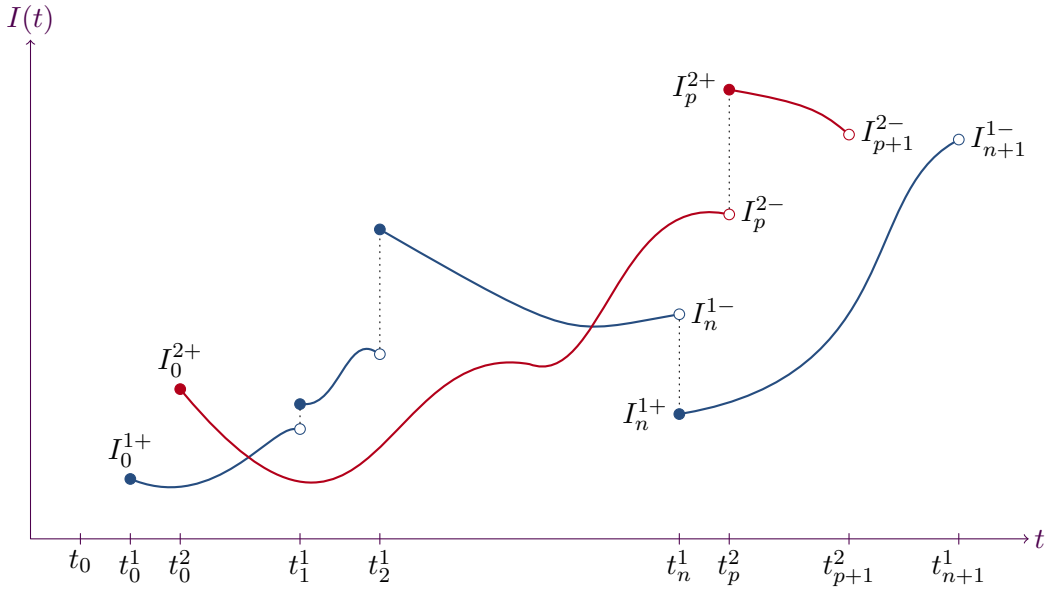
Tracking Investment Performance Amidst Sporadic and Uncorrelated Buy/Sell Transactions

Paul Kotschy

9 December 2025

Compiled on December 12, 2025



Abstract

AN IMPORTANT TASK[1] of an investment portfolio manager is to report on the performance of not only a portfolio as a whole, but also of its constituent investment instruments. For a portfolio comprised of a single instrument into which a single lump sum cash injection is made, reporting on performance is trivially easy. But life is seldom easy. Usually, multiple disparate instruments make up an investment portfolio, and cash injections and ejections are a mix of ad hoc lump sum and regular deposits and withdrawals. Under such a regime of sporadic and uncorrelated buy/sell transactions, reporting on performance is indeed not trivial.

Herein, then, I introduce two numerical measures of performance. The measures are specially taylored to account for such real-world investing regimes. The first measure, dubbed the *effective long growth rate*, accommodates cash injections and ejections on a set of investments of arbitrary size and at arbitrary times. The effective long growth rate is formulated as a composite of exponential growth rates. It is an inherently nonlinear measure, weighted by transaction size and by the lapsed time between adjacent transactions. The second measure, dubbed the *spot proportional change*, accounts for the deleterious effect of transaction costs on an investment's performance.

These two new performance measures have been incorporated in the author's PKINVEST portfolio management software.

[1] I declare this to be my own work, entirely. In particular, no AI was used in any research, analysis, synthesis, writing, nor typesetting of this work. In short, AI was not recruited at any time in this work. Errors and inaccuracies are therefore proudly my own.

Contents

1 Introduction 3

2 Analysis 4

2.1 Time-discontinuous regime for a single investment . . . 4

2.1.1 Transaction costs . . . 5

2.2 Effective long growth rate . . . 6

2.2.1 Dividends . . . 7

2.2.2 Numerical scheme . . . 8

2.3 Spot proportional change . . . 10

2.4 Time-discontinuous regime for an investment portfolio . . . 11

3 Implementation 13

3.1 The Model component . . . 13

3.1.1 The `create.sql` file . . . 13

3.1.2 The `demo.sql` file . . . 14

3.2 The View component . . . 15

3.2.1 The `index.html-tmpl` file . . . 15

3.2.2 The `pkinvest.css` file . . . 15

3.2.3 The `login.css` file . . . 22

3.2.4 The `portfolio2.m4-tmpl` file . . . 23

3.3 The Controller component . . . 25

3.3.1 The `m7config.h` file . . . 25

4 Acknowledgments 27

1 Introduction

THE NEEDS OF A PRUDENT PRIVATE INVESTOR are similar to those of an institutional investor. Broadly, a prudent private investor seeks both to maximise return and to minimise risk of loss on his or her suite of investments over time. To satisfy these two objectives, the private investor must:

1. *Analyse the quality* of actual and prospective investment instruments.

   The focus of the analysis must not be on past performances, but on forecasted future ones. Also, the instruments' contribution to the overall make-up of the portfolio must be considered.

2. *Construct a portfolio.*

   The portfolio must stand the best chance of exploiting economic and business fluctuations. The portfolio must be appropriately diversified. That is, not only must the response of the portfolio's member instruments to such fluctuations be uncorrelated, but there must also be sufficient redundancy between the instruments. The redundancy helps reduce the impact of failure of any one.

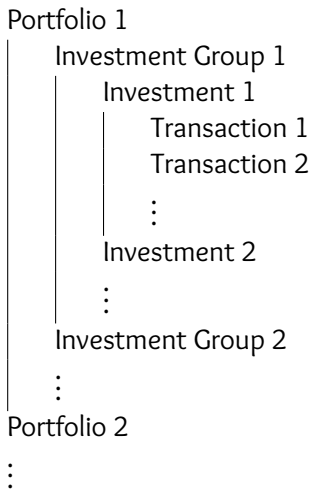
3. *Account for the costs* associated with each investment.

   It is known that for collective investment schemes, such as mutual funds, the expensiveness of a scheme is a good predictor of its performance.[1]

4. *Maintain records.*

   Records must be kept of buy/sell transactions, dividend and interest receipts, investment values, and so on. These records are needed to characterise the evolution of the individual investments, the investment groups, and the portfolio as a whole.

Using the author's PKINVEST software platform, a portfolio is constructed as a set of investment groups. In turn, each group is a set of investments. And to each investment is associated a sequence of transactions. This hierarchy is shown schematically as follows:



The contributions from each transaction to an investment, and from an investment to its group, and from a group to its portfolio, are all accounted for.

The first new performance measure introduced here is dubbed the *effective long growth rate*. It is computed for each investment, for each investment group, and for each portfolio. It accommodates the time-discontinuous evolution of a real investment portfolio. That is, it accommodates an investor making cash injections into the set of investments of arbitrary size and at arbitrary times. The effective long growth rate is formulated as a composite of exponential growth rates. It is an inherently non-linear measure, weighted by transaction size and by the time lapse between adjacent transactions.

The second new performance measure introduced here is dubbed the *spot proportional change*. It is introduced to account for the deleterious effect of investment transaction costs on the spot growth performance of an investment.

In the ensuing Section 2, a detailed mathematical analysis of a typical real investment regime is given, with particular emphasis on the development of investment performance measures. Section 3 addresses aspects of the software design and implementation of PKINVEST. The two new performance measure have been incorporated into PKINVEST.

2 Analysis

INVESTMENT TRANSACTIONS conducted by private investors are typically lump sum cash injections of arbitrary size, made at arbitrary times, and into a variety of investments. Unfortunately, not only does this ad hoc and uncorrelated behaviour obscure the intrinsic growth of the individual investments, but it also hampers an effective comparison between the investments.

Therefore, to help expose intrinsic growth and to foster such comparisons, a mathematical analysis must from the outset account for the presence of this ad hoc and uncorrelated behaviour. To begin, we appeal to the following definitions:

L_n — Lump sum cash injection into an investment at time t_n.

I — Total value of the investment I at time t.

I_n^- — Total value of the investment I at time t_n, but *just before* the lump sum cash injection L_n. That is $I_n^- = I(t_n^-)$.

I_n^+ — Total value of the investment I at time t_n, but *just after* the lump sum cash injection L_n. That is $I_n^+ = I(t_n^+)$.

q — Number of units of investment I held by the investor.

i — Investment value per unit of investment at time t. That is $i(t) = I(t)/q(t)$.

c — Cost of conducting an investment transaction, measured as a fraction of the absolute difference $|I_n^+ - I_n^-|$, and assumed to be a constant fraction. Note that cash injections and cash ejections both incur a positive transaction cost. And c being strictly positive reflects this.

D_n — Value of a dividend received over the $[t_n, t_{n+1})$ time interval and pertaining to investment I.

k — *Effective long growth rate* for investment I calculated at time t.

g_n — *Spot proportional change* for the $[t_{n-1}, t_n]$ time interval.

P — Total value of the investment portfolio at time t.

K — *Effective long growth rate* for the investment portfolio as a whole, calculated at time t.

2.1 Time-discontinuous regime for a single investment

A typical evolution of a single investment I is shown schematically in Figure 1. A transaction on the investment is simply either a lump sum cash injection into the investment or a cash ejection from it. Each transaction results in jump discontinuity in the evolution at the time of the transaction.

The equation for the time-continuous evolution of investment I over the $[t_n, t_{n+1})$ time interval is

$$I(t) = I_n^+ e^{\int_{t_n}^{t} \kappa(t)\,\mathrm{d}t} \qquad \begin{array}{l} \text{for } t \in [t_n, t_{n+1}),\ n = 0, 1, 2, \ldots \\ \text{and } I_0^+ \equiv I_0 \end{array} \tag{1}$$

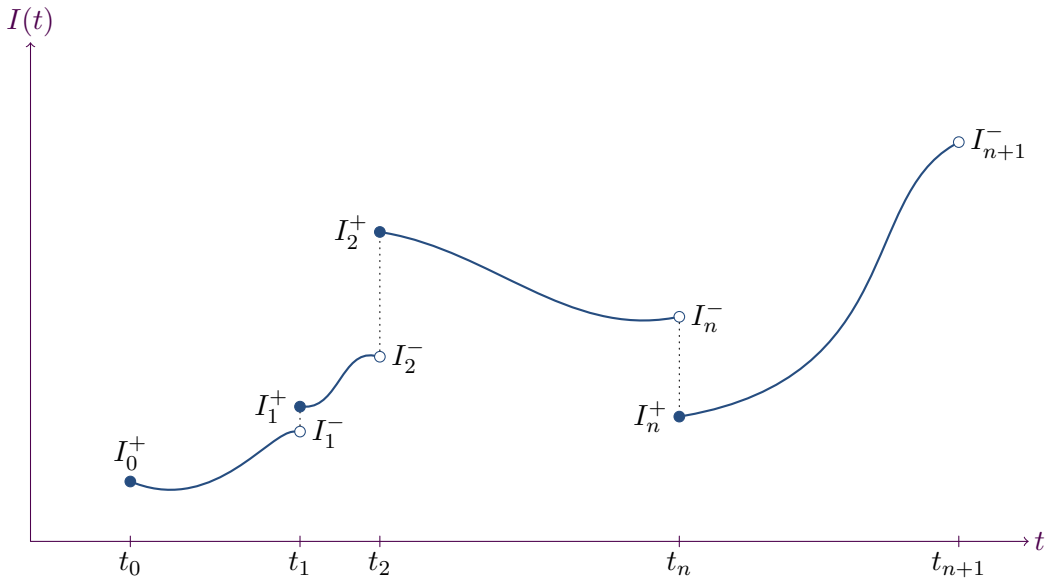


Figure 1: A typical single investment regime showing its ad hoc and time-discontinuous character. As indicated, cash injections and ejections are of arbitrary size and are made at arbitrary times.

where $\kappa(t)$—not to be confused with $k(t)$—is the instantaneous time rate of relative change in I at time t. Equation (1) is easily derived, as follows. The infinitesimal change in the value of the investment at any time t is

$$\mathrm{d}I(t) = I(t)\kappa(t)\,\mathrm{d}t$$

so that

$$\frac{\mathrm{d}I(t)}{\mathrm{d}t} = \kappa(t)I(t)$$

This is a simple linear homogenous differential equation. By dividing both sides by $I(t)$ and integrating, the solution (1) is obtained.

Equation (1) suggests that a reasonable definition for a time-continuous growth rate which is constant over the $[t_n, t_{n+1})$ interval is

$$k_n = \langle\kappa(t)\rangle|_{t_n}^{t_{n+1}} = \frac{1}{t_{n+1}-t_n}\int_{t_n}^{t_{n+1}} \kappa(t)\,\mathrm{d}t$$

Obviously, k_n is the time-average of $\kappa(t)$ over the $[t_n, t_{n+1})$ interval. The time-continuous investment evolution equation (1) may then be approximated as

$$\boxed{I(t) \approx I_n^+ e^{k_n(t-t_n)} \qquad \begin{array}{l}\text{for } t \in [t_n, t_{n+1}),\ n = 0, 1, 2, \ldots \\ \text{and } I_0^+ \equiv I_0\end{array}} \tag{2}$$

2.1.1 Transaction costs

Since a lump sum cash injection into an investment, or a cash ejection from it, introduces a jump discontinuity in the time evolution of the investment's value, it is easy to state the governing equation for the n-th transaction at time t_n as

$$I_n^+ = I_n^- + L_n - c\left|I_n^+ - I_n^-\right|$$

It is also easy to prove from this governing equation that

$$I_n^+ = I_n^- + L_n - \mathrm{sgn}(L)c(I_n^+ - I_n^-)$$

from which we obtain

$$\boxed{I_n^+ = I_n^- + \frac{L}{1+\mathrm{sgn}(L)c}} \tag{3}$$

Equations (2) and (3) allow the influence of the set of lump sum cash injections $\{L_n \,|\, n = 0, 1, 2, \dots\}$ to be quantified. If only a single lump sum injection is made at time t_0, say, then an effective long growth rate constant k could be specified by (2) with k identified with k_0. However, whereas (2) pertains to the time evolution of I_0^+, the investor initially "sacrificed" an amount $L_0 = (1+c)I_0$, which amount is greater than I_0^+. And although the transaction fractional cost c might be small relative to I_0^+, it may not be small relative to k. Therefore, (2) and (3) suggest that a better effective long growth rate constant could be specified implicitly by

$$I(t) = L_0 e^{k(t-t_0)} \qquad \text{for } t \in [t_0, t_1) \tag{4}$$

so that

$$k = k(t) = \frac{1}{t - t_0} \ln\left(\frac{I(t)}{L_0}\right) \qquad \text{for } t \in [t_0, t_1)$$

Finally, as a matter of record keeping, investment transactions usually involve the exchange of a number of units of the investment instrument, with each unit having an agreed value. That is, $I(t) = q(t)i(t)$, so that

$$\boxed{k(t) = \frac{1}{t - t_0} \ln\left(\frac{q(t)i(t)}{L_0}\right) \qquad \text{for } t \in [t_0, t_1)} \tag{5}$$

2.2 Effective long growth rate

A naïve interpretation of (5) is that an investment instrument's real growth over time is exponential. But that interpretation is incorrect. Equity investments, for example, are subject to "external forces", and these forces may have political, macroeconomic or psychological origins. Rather, the purpose of (4) and (5) is: 1. to provide a long-term investor with a sensible average growth rate constant for investment I; 2. to enable the investor to track the evolution of the rate constant; and 3. to place comparisons between independent investments on a firmer quantitative footing.

Equation (5) applies only to a single transaction at time t_0. We must extend the analysis to obtain an effective long growth rate constant applicable in an investment regime involving multiple transactions of arbitrary size and conducted at arbitrary times, as shown in Figure 1 on page 5.

Applying (2) to the $[t_1, t_2)$ time interval gives

$$\begin{aligned}
I(t) &= I_1^+ e^{k_1(t-t_1)} \qquad \text{for } t \in [t_1, t_2) \\
&= \left(I_1^- + \frac{L_1}{1 + \text{sgn}(L_1)c}\right) e^{k_1(t-t_1)} \qquad \text{from (3)} \\
&= \left(I_0^+ e^{k_0(t_1-t_0)} + \frac{L_1}{1 + \text{sgn}(L_1)c}\right) e^{k_1(t-t_1)} \\
&= \left(\frac{L_0}{1 + \text{sgn}(L_0)c} e^{k_0(t_1-t_0)} + \frac{L_1}{1 + \text{sgn}(L_1)c}\right) e^{k_1(t-t_1)} \\
&= \frac{L_0}{1 + \text{sgn}(L_0)c} e^{k_0(t_1-t_0)+k_1(t-t_1)} + \frac{L_1}{1 + \text{sgn}(L_1)c} e^{k_1(t-t_1)}
\end{aligned}$$

This suggests that for the $[t_1, t_2)$ interval, a choice for an effective long growth rate constant could be defined implicitly by the equation

$$I(t) = \frac{L_0}{1 + \text{sgn}(L_0)c} e^{k(t-t_0)} + \frac{L_1}{1 + \text{sgn}(L_1)c} e^{k(t-t_1)} \qquad \text{for } t \in [t_1, t_2)$$

However, as argued earlier, k here is a post-transaction cost measure. Specifically, $L_0/(1 + \text{sgn}(L_0)c) < L_0$ and $L_1/(1 + \text{sgn}(L_1)c) < L_1$, and L_0 and L_1 are the actual cash flows made by the investor. Therefore, following (4), a better long growth rate constant which accounts for the effect of transaction costs could be defined by

$$I(t) = L_0 e^{k(t-t_0)} + L_1 e^{k(t-t_1)} \qquad \text{for } t \in [t_1, t_2)$$

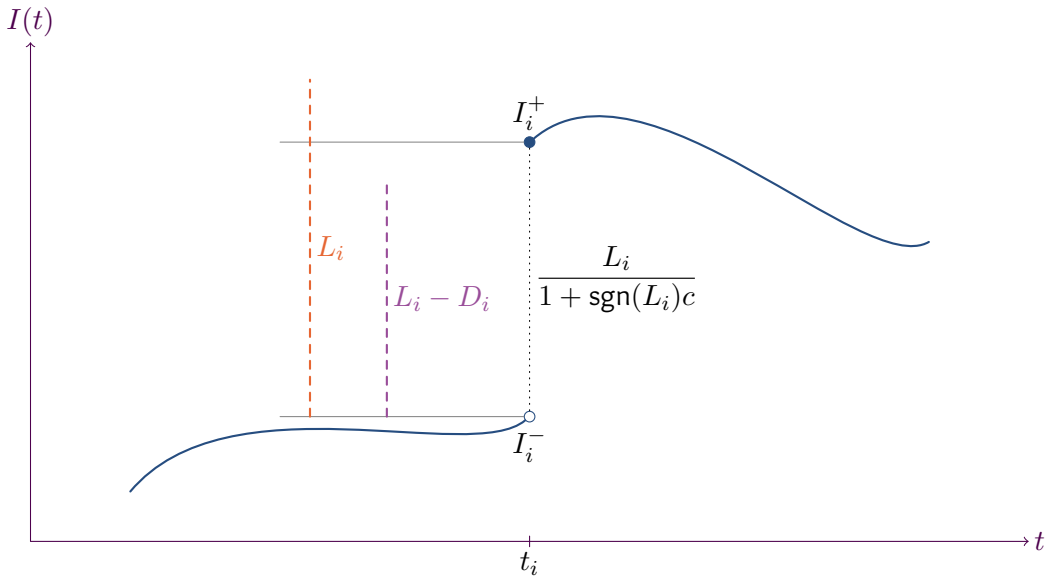


Figure 2: Schematical representation of the magnitude of the lump sum L_i, the effect of the receipt of the dividend D_i, and the actual jump in the investment value as a result of the transaction at time t_i.

And once again, as a matter or record keeping, since $I(t) = q(t)i(t)$, $k(t)$ is defined implicitly by

$$L_0 e^{k(t)(t-t_0)} + L_1 e^{k(t)(t-t_1)} = q(t)i(t) \qquad \text{for } t \in [t_1, t_2) \tag{6}$$

Since L_0 and L_1 would have been recorded at the transaction times t_0 and t_1, (6) may be solved for $k(t)$ whenever $t_1 \leqslant t < t_2$.

The analysis for the $[t_2, t_3)$ time interval and for subsequent time intervals follows analogously. And so, for any $[t_n, t_{n+1})$ interval, we may implicitly define a pre-transaction-cost long growth rate constant $k(t)$ by

$$\sum_{i=0}^{n} L_i e^{k(t)(t-t_i)} = q(t)i(t) \qquad \text{for } t \in [t_n, t_{n+1}),\ n = 0, 1, 2, \ldots \tag{7}$$

2.2.1 Dividends

As a holder of a portion of a company's outstanding shares, an investor is entitled to receive dividends from the company. A dividend is paid by the company into the investor's banking account or share trading account. If it may be assumed that the dividends receivable over the $[t_i, t_{i+1})$ interval are in fact all received at the time of the i-th transaction, then the net effect is to reduce the investor's required cash for the i-th transaction, as shown in Figure 2 above. By reducing the investor's required cash for the i-th transaction, the effect of receipt of a dividend is to increase the investment's effective long growth rate, $k(t)$.

In the absence of receipt of any dividends, the solution to (7) for $k(t)$ is sufficient. But because the dividend effectively discounts the investor's lump sum, (7) must be modified to give

$$\boxed{\sum_{i=0}^{n} (L_i - D_i) e^{k(t)(t-t_i)} = q(t)i(t) \qquad \text{for } t \in [t_n, t_{n+1}),\ n = 0, 1, 2, \ldots} \tag{8}$$

Unfortunately, (8) cannot be solved analytically, and a simple closed-form expression for the effective long growth rate cannot therefore be obtained. We are obliged to appeal to a numerical approximation for $k(t)$.

2.2.2 Numerical scheme

Let N be the number of buy/sell transactions. Then the set of transaction times is $\{t_0, t_1, \ldots, t_{N-1}\}$, and we wish typically to obtain a value for $k(t)$ for some $t \geqslant t_{N-1}$.

A simple "predictor–corrector" scheme is suggested by (8), using

$$k(t) = \frac{1}{t - t_m} \ln\left(\frac{q(t)i(t) - \sum_{i=0, i\neq m}^{N-1}(L_i - D_i)e^{k(t)(t-t_i)}}{L_m - D_m}\right) \quad \text{for some } m \text{ and } t \geqslant t_{N-1} \tag{9}$$

But (9) is not well posed for the special case of $m = N - 1$ and $t = t_{N-1}$. So we must look elsewhere.

An alternative numerical scheme which will work in that special case is the well-known Bisection Method.[2] The method will always converge to a solution, albeit "slowly".

Rewrite equation (8) as

$$F(k(t)) = \sum_{i=0}^{N-1}(L_i - D_i)e^{k(t)(t-t_i)} - q(t)i(t) = 0 \qquad \text{for } t \geqslant t_{N-1},\ N = 1, 2, \ldots \tag{10}$$

Then the solution to $k(t)$ is a root of F.

The Bisection Method requires that an interval be supplied which is known to contain a desired solution. The interval is specified with a lower bounding value and an upper bounding value. We seek an initial approximation to $k(t)$ which can be used as one of these bounding values.

First bounding value. Suppose that instead of seeking a long growth constant over the full $[t_0, t]$ interval, we assign each interval its own long growth constant. Then from (2)

$$q(t_{i+1}^-)i(t_{i+1}^-) = q(t_i^+)i(t_{i+1}) = q(t_i^+)i(t_i)e^{k_i(t_{i+1}-t_i)}$$

since the number of investment units are constant over each $[t_i, t_{i+1})$ interval. So

$$k_i = \frac{1}{t_{i+1} - t_i} \ln\left(\frac{i(t_{i+1})}{i(t_i)}\right) \tag{11}$$

We are able to easily compute the set $\{k_i\}$. Can we therefore specify a suitable initial approximation to $k(t)$ using $\{k_i\}$? For $t \in [t_0, t_1)$

$$q(t)i(t) = q_0^+ i_0 e^{k_0(t-t_0)} = \frac{L_0 - D_0}{1 + c}e^{k_0(t-t_0)} \qquad \text{(using (2.1.1))}$$

And for $t \in [t_1, t_2)$

$$\begin{aligned} q(t)i(t) &= q_1^+ i_1 e^{k_1(t-t_1)} = (q_1^- + \Delta q_1)i_1 e^{k_1(t-t_1)} \\ &= (q_0^+ i_1 + \Delta q_1 i_1)e^{k_1(t-t_1)} \\ &= (q_0^+ i_0 e^{k_0(t_1-t_0)} + \Delta q_1 i_1)e^{k_1(t-t_1)} \\ &= \frac{1}{1 + c}\left((L_0 - D_0)e^{k_0(t_1-t_0)} + (L_1 - D_1)\right)e^{k_1(t-t_1)} \end{aligned}$$

And so on. Therefore, for any $N = 1, 2, \ldots$

$$q(t)i(t) = \frac{1}{1 + c}\sum_{i=0}^{N-1}(L_i - D_i)e^{\sum_{j=i}^{N-2} k_j(t_{j+1}-t_j)}e^{k_{N-1}(t-t_{N-1})} \qquad \text{for } t \in [t_{N-1}, t_N) \tag{12}$$

Comparison of (12) with (8) suggests that a suitable initial approximation to $k(t)$ is given by

$$e^{k(t)(t-t_0)} = \frac{1}{1 + c}e^{\sum_{j=0}^{N-2} k_j(t_{j+1}-t_j) + k_{N-1}(t-t_{N-1})}$$

from which

$$k(t) = \frac{1}{t - t_0} \ln \left(\frac{1}{1+c} e^{\sum_{j=0}^{N-2} k_j (t_{j+1} - t_j) + k_{N-1}(t - t_{N-1})} \right)$$

This approximation can be simplified by appealing to the definition of the $\{k_i\}$ in (11). Consider the summation in the exponential:

$$\sum_{j=0}^{N-2} k_j (t_{j+1} - t_j) + k_{N-1}(t - t_{N-1}) = \sum_{j=0}^{N-2} \ln(\frac{i_{j+1}}{i_j}) + \frac{t - t_{N-1}}{t_N - t_{N-1}} \ln(\frac{i_N}{i_{N-1}}) \tag{13}$$

But

$$\begin{aligned}
\sum_{j=0}^{N-2} \ln(\frac{i_{j+1}}{i_j}) &= \sum_{j=0}^{N-2} (\ln i_{j+1} - \ln i_j) \\
&= \ln i_{N-1} + \sum_{j=0}^{N-3} \ln i_{j+1} - \ln i_0 - \sum_{j=1}^{N-2} \ln i_j \\
&= \ln i_{N-1} - \ln i_0 + \sum_{j=0}^{N-3} \ln i_{j+1} - \sum_{j=0}^{N-3} \ln i_{j+1} \\
&= \ln(\frac{i_{N-1}}{i_0})
\end{aligned}$$

The summation simplifies to

$$\sum_{j=0}^{N-2} k_j (t_{j+1} - t_j) + k_{N-1}(t - t_{N-1}) \;=\; \ln(\frac{i_{N-1}}{i_0}) \;+\; \frac{t - t_{N-1}}{t_N - t_{N-1}} \ln(\frac{i_N}{i_{N-1}})$$

And the initial approximation to $k(t)$ then is

$$k(t) = \frac{1}{t - t_0} \ln \left(\frac{1}{1+c} \left(\frac{i_{N-1}}{i_0} \right) \left(\frac{i_N}{i_{N-1}} \right)^{\frac{t - t_{N-1}}{t_N - t_{N-1}}} \right) \qquad \text{for } t \in [t_{N-1}, t_N),\ N = 1, 2, \ldots \tag{14}$$

Application of (14) assumes the existence of the $[t_{N-1}, t_N)$ time interval for which $i(t_N)$ is known. But if it is not known, then we are inclined to consider $i(t_N)$ for t_N vanishingly close to t^+. And since

$$\lim_{t_N \to t^+} \left(\frac{t - t_{N-1}}{t_N - t_{N-1}} \right) = 1 \qquad \text{for } t > t_{N-1}$$

we have

$$k(t) = \frac{1}{t - t_0} \ln \left(\frac{1}{1+c} \cdot \frac{i(t)}{i_0} \right)$$

On the other hand, if we are interested in $k(t_{N-1})$, then set $t = t_{N-1}$ in (14) to give

$$k(t_N) = \frac{1}{t_{N-1} - t_0} \ln \left(\frac{1}{1+c} \cdot \frac{i(t_{N-1})}{i_0} \right)$$

But because these two expressions have the same functional form, we are free to drop recourse to $i(t_N)$, and set as an initial approximation to $k(t)$

$$\boxed{k(t) = \frac{1}{t - t_0} \ln \left(\frac{1}{1+c} \cdot \frac{i(t)}{i_0} \right) \qquad \text{for all } t \geqslant t_0} \tag{15}$$

Second bounding value. A second bounding value is obtained by making increasingly large "jumps" away from the first bounding value, as follows:

$k' \Leftarrow k(t)$ using (15). k' is the first bounding value.
$l \Leftarrow 0$
$k'' \Leftarrow k' - (-\alpha)d$ for some $\alpha > 1$ and $d > 0$.
while $F(k')F(k'') > 0$ **do**
  $l++$
  $k'' \Leftarrow k' - (-\alpha)^l d$
k'' is now a second bounding value.
Set k' and k'' such that $k'' > k'$.

Inspection of the algorithm suggests that an improvement in efficiency may be obtained by removing the "$(-\alpha)^l$" exponent computation. Let k''_l be the l-th iterative attempt at finding a second bounding value k''. Then

$$k''_l = k' - (-\alpha)^l d \quad \text{and} \quad k''_{l-1} = k' - (-\alpha)^{l-1} d$$

So

$$\begin{aligned} k''_l &= k' - (-\alpha)(-\alpha)^{l-1} d \\ &= k' - (-\alpha)[k' - k''_{l-1}] \\ &= (1+\alpha)k' - \alpha k''_{l-1}, \qquad \text{for } l = 1, 2, 3, \ldots, \text{ with } k''_0 = k' - d \end{aligned}$$

Therefore the revised algorithm is as listed in Algorithm 1.

Algorithm 1 Compute a $[k', k'']$ bounding interval for the Bisection Method.

$k' \Leftarrow k(t)$ using (15). k' is the first bounding value.
$k'' \Leftarrow k' - d$ for some $d > 0$.
while $F(k')F(k'') > 0$ **do**
  $k'' \Leftarrow (1+\alpha)k' - \alpha k''$ for some $\alpha > 1$.
k'' is now a second bounding value.
Set k' and k'' such that $k'' > k'$.

Of course, the actual implementation of Algorithm 1 must account for the possibilities of $F(k') = 0$, $F(k'') = 0$, and of never finding a k''.

Bisection Method. With a $[k', k'']$ bounding interval now known, the essential elements of the Bisection Method are listed in Algorithm 2.

Algorithm 2 Essential elements of the Bisection Method.

Compute k' and k'' using Algorithm 1.
while $|k'' - k'| > \epsilon$ for some small ϵ **do**
  $k \Leftarrow \frac{1}{2}(k' + k'')$
  if $F(k')F(k) < 0$ **then**
    $k'' \Leftarrow k$
  else
    $k' \Leftarrow k$

2.3 Spot proportional change

Whereas the effective long growth rate (§2.2) is defined over a time interval spanning multiple buy/sell transactions, the *spot proportional change* defined in this section concerns changes between two adjacent transactions, and involves only one (possibly nil) lump sum cash injection and one (possibly nil) dividend receipt.

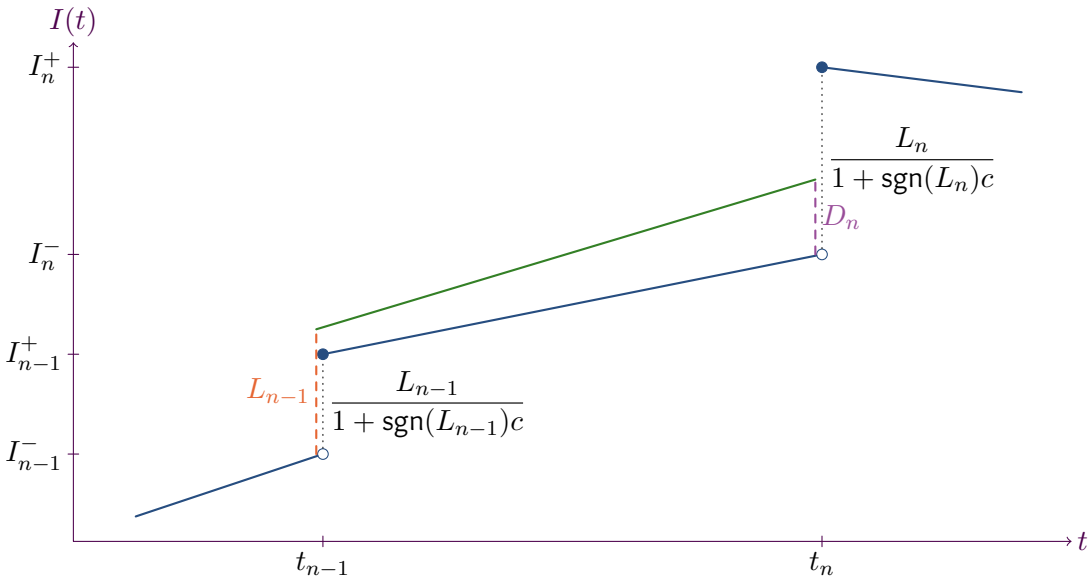


Figure 3: Idealisation of an investment's price history over the $[t_{n-1}, t_n)$ time interval. The endpoints of the **green line** are used to define the *spot proportional change* for the $[t_{n-1}, t_n]$ time interval.

The deleterious effect of transaction costs on the spot growth performance of an investment are often overlooked. I surmise this is because, firstly, the transaction costs are usually small relative to the overall size of a transaction. And secondly, it is not in the interests of the financial management community to disclose these costs. The notion of an spot proportional change formulated here accounts not only for the effect of these transactional costs, but also for the benefit of the receipt of dividends.

An idealisation of an investment's price history over the $[t_{n-1}, t_n]$ time interval is shown in Figure 3. Although the investment's actual value follows the trajectory shown in **blue**, the spot proportional change, g_n, for the $[t_{n-1}, t_n]$ time interval is identified as the relative difference between the endpoints of the **green line**. The defining equation is

$$g_n \equiv \frac{(I_n^- + D_n) - (I_{n-1}^- + L_{n-1})}{I_{n-1}^- + L_{n-1}} \quad \text{for } [t_{n-1}, t_n] \tag{16}$$

Why define the spot proportional change in this way? At time t_{n-1}, the investor makes an actual lump sum cash injection of L_{n-1}. So to ensure that the investment growth over the $[t_{n-1}, t_n]$ interval not be artificially inflated by this lump sum, we must begin with $(I_{n-1}^- + L_{n-1})$ and not simply I_{n-1}. Next, at the end of the interval, the investor receives a dividend D_n. Provided it is reinvested, it will serve to increase the value of the investment, and so should be factored into a measure of spot growth.

We wish to express g_n in terms of quantities which are recorded for each buy/sell transaction. By definition

$$I_n^- = q(t_n^-)i(t_n^-) = q(t_{n-1}^+)i(t_n) \equiv q_{n-1}i_n$$

And defining

$$\Delta q_n = q(t_n^+) - q(t_n^-) = q(t_n^+) - q(t_{n-1}^+) = q_n - q_{n-1}$$

it is easy to show that

$$\boxed{g_n = \frac{(q_n - \Delta q_n)i_n - (q_n - \Delta q_n - \Delta q_{n-1})i_{n-1} + D_n - L_{n-1}}{(q_n - \Delta q_n - \Delta q_{n-1})i_{n-1} + L_{n-1}} \qquad \text{for } n = 1, 2, 3, \ldots} \tag{17}$$

2.4 Time-discontinuous regime for an investment portfolio

The numerical schemes listed in Algorithms 1 and 2 are for a single investment into which lump sum injections are made. However, in reality, a private investor must manage multiple investments. The analysis must therefore be extended to account for this.

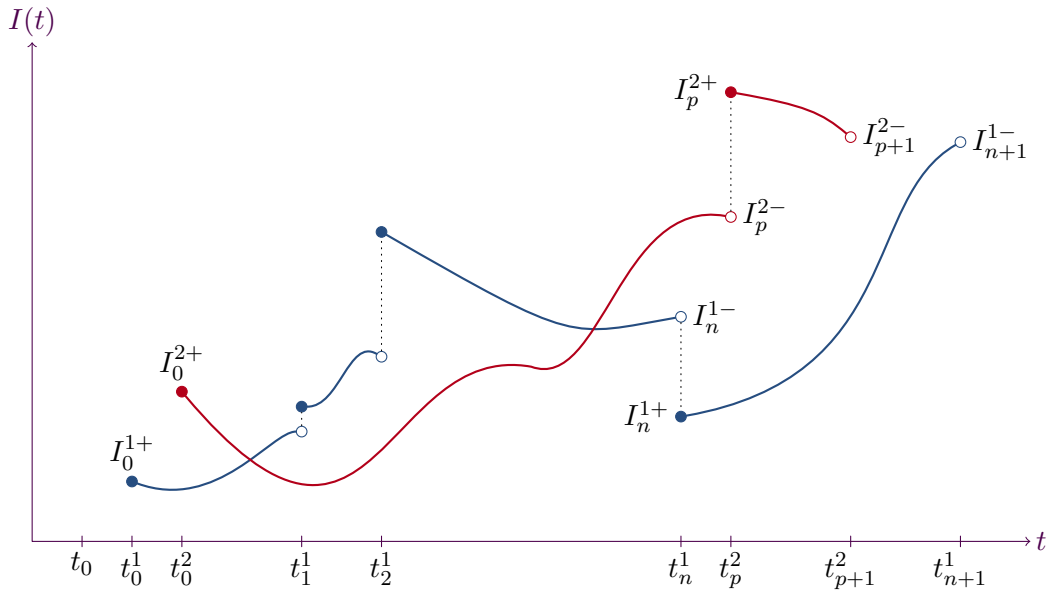


Figure 4: A typical portfolio comprising two investments I^1 and I^2.

Suppose an investor's portfolio consisted of only two investments. A typical set of transactions for the two investments is shown schematically in Figure 4.

Even for a simple portfolio consisting of no more than two investments, the calculation of an aggregate growth rate for the portfolio as a whole is not trivial. Indeed, as depicted in Figure 4, two investments in the portfolio need not even share an initial investment time.

The simple question "How is my portfolio performing?" belies the difficulty it presents in providing a sensible answer. To provide an answer, we must extend the analysis of the growth of a single investment to that of the portfolio as a whole.

Suppose that the investor's portfolio consists of M independent investments, with the m-th investment having the set $\{t_0^m, t_1^m, \ldots, t_{N^m-1}^m\}$ of N^m transaction times. Then, applying (8) to the m-th investment, we may write

$$\sum_{i=0}^{N^m-1} (L_i^m - D_i^m) e^{k^m(t)(t-t_i^m)} = q^m(t) i^m(t) \qquad \text{for } m = 1, 2, \ldots, M \text{ and } t \geqslant \max\left\{t_{N^m-1}^m \,|\, m = 1, 2, \ldots, M\right\} \tag{18}$$

and where L_i^m is a lump sum cash injection into the m-th investment at time t_i^m, D_i^m is the (possibly nil) dividend extracted from the m-th investment at t_i^m.

The value of the investor's portfolio at time t is obviously

$$P(t) = \sum_{m=1}^{M} q^m(t) i^m(t) \tag{19}$$

Scrutiny of (18) and (19), together with (10), suggests that a pre-transaction-cost effective long growth rate, $K(t)$, for the portfolio as a whole may be computed as the root of the equation

$$F(K(t)) = \sum_{m=1}^{M} \sum_{i=0}^{N^m-1} (L_i^m - D_i^m) e^{K(t)(t-t_i^m)} - P(t) = 0 \qquad \text{for } t \geqslant \max\left\{t_{N^m-1}^m \,|\, m = 1, 2, \ldots, M\right\} \tag{20}$$

The Bisection Method described in Section 2.2.2 may then be used to compute $K(t)$.

3 Implementation

The *effective long growth rate* and the *spot proportional change* were introduced in Section 2 as two new numerical measures of the performance of an investment portfolio. Both measures have been been incorporated in the author's PKINVEST portfolio management software.

PKINVEST is an online application. This means that it is constructed as a Web-based software application. Its software design architecture adheres to the well-known Model-View-Controller (MVC) design pattern.[3] The "Model" component is the data stored in an SQLite3 SQL database. The "View" component is a set of M4-formatted files which get transformed "on-the-fly" into a set of HTML files, and which in turn are rendered in a user's Web browser. The "Controller" is a set of C source code files which compile to create the `pkinvest` binary. The binary controls the integration of the data with the M4 files during their transformation into HTML.

My M7 programming framework plays a central role in facilitating this transformation. The framework consists of a set of predefined M4 macros and an application programming interface. The macros reside in the `/usr/local/m7/defs/html/stdlib2.m4` filepath. In the context of Web programming, the macros provide the Web developer with a layer of abstraction over HTML, allowing Web content to be prepared with M4-formatted files rather than with HTML-formatted files.[2] The abstraction helps the developer maintain uniformity and consistency in the derived HTML. This is because the expansion of a macro, via an implicit invocation of the `m4` macro-processor, remains the same whenever and wherever the macro is called.

The directory structure of the code base has been organised, in part, to reflect adherence to the abovementioned MVC design pattern. For example, the `src/` directory contains the "`.h`" and "`.c`" C source code files; and the `m4/` directory contains a set of "`.m4-tmpl`" files which are compiled into the abovementioned "`.m4`" M4 files prior to the installation of PKINVEST.

In this section, the application of the MVC design pattern in the implemention of PKINVEST is presented on a file by file basis. The ensuing documentation in this section was prepared with the help of the my LaTeX-based PKTECHDOC[4] documentation system. Using PKTECHDOC, it was possible to annotate the source code elements directly in their respective source files, but to have the annotations be typeset in this document using LaTeX.

3.1 The Model component

3.1.1 The `create.sql` file

The database schema for PKINVEST is intentionally simple. It constists of the `Portfolio` SQL table, the `InvestmentGroup` table, the `Investment` table, and the `Transactn` table.

Conceptually, a portfolio comprises a set of investment groups, such as asset classes. An investment group comprises a set of investments, such as a set of individual equities. And an investment comprises a set of transactions, such as the purchase of additional shares in an equity, the extraction of a dividend, or the preparation of a time snapshot of the value of an investment.

Therefore, as is reflected in some of the integer fields, changing `Portfolio` affects `InvestmentGroup`, changing `InvestmentGroup` affects `Investment`, and changing `Investment` `Investment` affects `Transactn`. For example, an entry cannot be deleted from the `Portfolio` table if at least one entry in `InvestmentGroup` refers to it. And an entry cannot be deleted from the `Investment` table if at least one entry in `Transactn` refers to it.

```
1  CREATE TABLE Portfolio (
2     id integer not null primary key,
3     name text
4  );
5  CREATE TABLE InvestmentGroup (
```

[2] M4 is a macro language processor. The `m4` binary utility program is found on all UNIX compute systems.

```
    id integer not null primary key,
    portfolio integer not null,
    name text
);
CREATE TABLE Investment (
    id integer not null primary key,
    investmentGroup integer not null,
    name text,
    code text
);
CREATE TABLE Transactn (
    id integer not null primary key,
    investment integer not null,
    epochMins biginteger not null,
    buySell float not null,
    quantity integer non null,
    unitSpotValue float not null,
    dividend float not null,
    note text
);
```

3.1.2 The `demo.sql` file

This file contains a set of SQL `INSERT` instructions used to populate the `demo.db` SQLite3 database.

```
INSERT INTO "Portfolio" VALUES(1,'Live');
INSERT INTO "Portfolio" VALUES(2,'Expired');
INSERT INTO "Portfolio" VALUES(3,'Experimental');

INSERT INTO "InvestmentGroup" VALUES(1,1,'Individual Equity');
INSERT INTO "InvestmentGroup" VALUES(2,1,'Local Collective Equity');
INSERT INTO "InvestmentGroup" VALUES(3,1,'Listed Property');
INSERT INTO "InvestmentGroup" VALUES(4,1,'Offshore');
INSERT INTO "InvestmentGroup" VALUES(5,1,'Bond');
INSERT INTO "InvestmentGroup" VALUES(6,1,'Cash');
INSERT INTO "InvestmentGroup" VALUES(7,2,'Expired');

INSERT INTO "Investment" VALUES(1,1,'Aveng','AEG');
INSERT INTO "Investment" VALUES(2,1,'BHP Billiton','BIL');
INSERT INTO "Investment" VALUES(3,1,'Kumba Iron Ore','KIO');
INSERT INTO "Investment" VALUES(4,1,'Santam','SNT');
INSERT INTO "Investment" VALUES(5,2,'Satrix Divi ETF','STXDIV');
INSERT INTO "Investment" VALUES(6,2,'Brantam: Shiraz','');
INSERT INTO "Investment" VALUES(7,6,'Current account','');
INSERT INTO "Investment" VALUES(8,6,'Coronation Money Market','CMMF');
INSERT INTO "Investment" VALUES(9,6,'Imara share trading account','');
INSERT INTO "Investment" VALUES(10,1,'Lewis','LEW');
INSERT INTO "Investment" VALUES(11,3,'Capital','CPL');
INSERT INTO "Investment" VALUES(12,3,'Redefine','RDF');
INSERT INTO "Investment" VALUES(13,3,'Proptrax ETF','PTXSPY');
INSERT INTO "Investment" VALUES(14,4,'DBX World ETF','DBXWD');
INSERT INTO "Investment" VALUES(15,6,'Cadiz Money Market','AFMM');

INSERT INTO "Transactn" VALUES(1,15,21563880,101.0,100,1.0,0.0,'');
INSERT INTO "Transactn" VALUES(2,15,21824520,101.0,100,1.0,0.0,'');
INSERT INTO "Transactn" VALUES(3,7,22041960,10.0,10,1.0,0.0,'');
INSERT INTO "Transactn" VALUES(4,9,21825960,202.0,200,1.0,0.0,'');
INSERT INTO "Transactn" VALUES(5,9,21872040,-101.0,-100,1.0,0.0,'');
INSERT INTO "Transactn" VALUES(6,1,21346440,103.0,10,10.0,0.0,'Getting some construction exposu
INSERT INTO "Transactn" VALUES(7,1,21913800,0.0,0,11.0,0.0,'Snapshot.');
```

```
36  INSERT INTO "Transactn" VALUES(8,2,21883560,45.0,200,0.222,0.0,'Billiton is trading at an attra
37  INSERT INTO "Transactn" VALUES(9,2,21902280,299.0,1300,0.22834,0.0,'');
38  INSERT INTO "Transactn" VALUES(10,2,21991560,0.0,0,0.24604,0.0,'Snapshot.');
```

3.2 The View component

3.2.1 The `index.html-tmpl` file

As a user's entry point into the PKINVEST application this file performs a simple HTML-based HTTP redirect in order that the `pkinvest` binary be executed.

```
1   <html>
2   <head>
3      <title>p k I n v e s t</title>
4      <meta http-equiv=Refresh content=0;
5            URL=https://__WEBSITENAME__/pkinvest-bin/pkinvest?req=dologin>
6   </head>
7   <!--
8   <body>
9      In __WEBSITENAME__/html/index.html
10  </body>
11  --!>
12  </html>
```

3.2.2 The `pkinvest.css` file

Various aspects of the visual appearance and layout of the PKINVEST's user-interface are controlled using Cascading Style Sheets (CSS).[3]

1. Styling of basic HTML elements.

```
1   /*
2    * Try #d6d6c2 for gold
3    */
4   body {
5      /*background-color: #369;*/
6      background-color: gray;
7      /*color: #c22;*/
8      /*color: #144;*/
```

[3] Learning about CSS, I found the following websites useful, in order:

- `http://www.w3.org/TR/REC-CSS1#classification-properties`
- `http://www.csszengarden.com/?cssfile=/163/163.css&page=0`
- `http://www.w3.org/Style/Examples/007/`
- `http://www.westciv.com/style_master/academy/css_tutorial/`
- `http://www.maujor.com/indexen.php`
- `http://html.tucows.com/`
- `http://css.maxdesign.com.au/listamatic/`
- `http://threeplusone.com/borders/`
- `http://www.alistapart.com/articles/practicalcss/`—Contains information on the use of `DIV`s with `FORM`s with `DIV`s.
- `http://www.mozilla.org/support/`—Contains information on cosmetics aspects, such as `button`s, and an input `form` field with rounded `DIV`s.

```
    color: black;
    font-family: sans-serif;
    margin: 0;
    border: 0;
}
a,a:link,a:visited {
    /*color: #c22;*/
    /*color: #144;*/
    color: black;
    text-decoration: none;
}
a:hover {
    /*color: #c22;*/
    background-color: #bbb;
    /*font-style: italic;*/
}
/*
 * 'a:active' and 'a.focus' are commented out.
 */
/*a:active {*/
/*   color: #fff;*/
/*   background-color: #888;*/
/*}*/
/*a.focus {*/
/*   color: #fff;*/
/*   background-color: #888;*/
/*}*/
table {
    /*width: auto;*/
    width: 100%;
    margin: 0;
    padding: 0;
    border: 0;
    empty-cells: inherit;
    border-collapse: collapse;
}
tr {
    vertical-align: top;
}
/*table tr td {*/
/*   border: 1px solid yellow;*/
/*}*/
/*p {*/
/*   padding: 5px;*/
/*}*/
select {
    font-size: 0.95em;
    /*color: #c22;*/
    /*color: #144;*/
    color: black;
    background-color: #ddd;
    border: 1px solid gray;
}
input,textarea {
    font-size: 0.95em;
    /*color: #c22;*/
    /*color: #144;*/
    color: black;
    background-color: #eee;
    padding: 1px 1px;
    border-top: 1px solid silver;
    border-left: 1px solid silver;
```

```
71      border-right: 1px solid gray;
72      border-bottom: 1px solid gray;
73   }
```

2. Class definitions for colours.

```
74   .blue {
75      color: #68b;
76   }
77   .dodgerblue {
78      color: dodgerblue4;
79   }
80   .peach {
81      color: #ffa;
82   }
83   .darkviolet {
84      color: darkviolet;
85   }
86   .red {
87      color: #c22;
88   }
89   .copper {
90      /*color: #e62;*/
91      color: #e92;
92   }
93   .brass {
94      color: #b5a642;
95   }
96   .bronze {
97      color: #a67d3d;
98   }
99   .steelblue {
100     /*color: #236b8e;*/
101     color: #26d;
102  }
103  .gold {
104     /*color: #acae74;*/
105     /*color: #a49e65;*/
106     color: #948a5c;
107  }
108  .newgold {
109     color: #cd7f32;
110  }
111  .lightgold {
112     /*color: #e4dec4;*/
113     color: #d4ceb4;
114  }
115  .silver {
116     color: silver;
117  }
```

3. General purpose style class definitions.

```
118  .small {
119     font-size: 0.85em;
120  }
121  .big {
122     font-size: 1.2em;
123  }
124  .huge {
125     font-size: 1.5em;
```

```
126   }
127   .bold {
128      font-weight: bold;
129   }
130   .emph {
131      font-style: italic;
132   }
133   .standout {
134      color: red;
135      font-size: 1.05em;
136   }
137   .message {
138      color: red;
139      /*font-style: italic;*/
140      /*font-size: 1.05em;*/
141   }
142   .textlabel {
143      text-align: left;
144      font-size: 0.85em;
145      font-style: italic;
146      /*white-space: nowrap;*/
147      /*padding-right: 0.5ex;*/
148   }
149
150   .textalignleft {
151      text-align: left;
152   }
153   .textalignright {
154      text-align: right;
155   }
156   .textaligncenter {
157      text-align: center;
158   }
159
160   .formsubmit {
161      /*color: #c22;*/
162      /*color: #144;*/
163      color: black;
164      background-color: #e4dec4;
165      padding: 0px 3px;
166      border-top: 1px solid silver;
167      border-left: 1px solid silver;
168      border-right: 1px solid gray;
169      border-bottom: 1px solid gray;
170   }
171   .formsubmit:hover {
172      /*background-color: #bbb;*/
173      background-color: #aaa;
174   }
```

4. Context-aware style class definitions.

```
175   .portfolioname {
176      /*font-weight: bold;*/
177      color: #e92;v        /* copper */
178      //color: #cd7f32;    /* newgold */
179   }
180
181   .investmentgroupname {
182      /*font-weight: bold;*/
183      /* steelblue */
```

```
    /*color: #236b8e;*/
    /*color: #26d;*/
    color: #c22; /* red */
}

.investmentname {
    /*font-weight: bold;*/
    /*color: darkviolet;*/
    color: green;
}

.transactionname {
    /*font-weight: bold;*/
    color: #13f; /* Blue */
}

.logoname {
    font-size: 1.3em;
    /*font-weight: bold;*/
    /*font-style: italic;*/
    text-decoration: none;
}
a:hover span.logoname {
    color: #d4ceb4;
    background-color: #eee;
}

ul.nakedul {
    list-style: none;
    display: inline;
    padding: 0;
    margin: 0;
    /*height: 10%;*/
}
ul.nakedul li {
    display: inline;
    padding: 0;
    margin: 0;
}

table.pkinvestpage {
    /* Fiddling begins. PJ Kotschy. 8Sep11. */
    /*margin: auto;*/
    /*width: 50%;*/
    /*width: 90ex;*/
    /*border: 2px solid #666;*/
    /* Fiddling ends. */
    border: 0;
}

table tr.bannerbar {
    background-color: #eee; /*#333;*/
    vertical-align: bottom;
}
table tr.bannerbar td {
    /*border-bottom: 1px solid silver;*/
    border-bottom: 1px solid gray;
}

table tr.modulebar td {
    background-color: #d4ceb4;
    font-size: 1.2em;
```

```
246      /*text-align: center;*/
247      text-align: left;
248      border-bottom: 1px solid #aaa;
249      padding: 3px 2px;
250   }
251   /*table tr.modulebar td ul li {*/
252   /*   padding-right: 2px;*/
253   /*   padding-left: 2px;*/
254   /*}*/
255   table tr.modulebar td ul li:before {
256      content: "|";
257   }
258   table tr.modulebar td ul li:first-child:before {
259      content: "";
260   }
261
262   table tr.actionbar {
263      background-color: silver;
264      border-top: 1px solid #aaa;
265      border-bottom: 1px solid gray;
266   }
267   /*table tr.actionlist {*/
268   /*   background-color: #ddd;*/
269   /*}*/
270
271   table tr td.workspace {
272      /*background-color: black;*/
273      background-color: #333;
274      color: #d4ceb4;
275      padding: 0.5ex;
276   }
277
278   table tr td.alignleft {
279      text-align: left;
280      /*padding-left: 0.4ex;*/
281      /*padding-right: 0.4ex;*/
282   }
283   table tr td.alignright {
284      text-align: right;
285      /*padding-left: 0.4ex;*/
286      /*padding-right: 0.4ex;*/
287   }
288   table tr td.aligncenter {
289      text-align: center;
290   }
291
292   table.daysofmonth {
293      width: auto;
294   }
295   table.daysofmonth tr td {
296      padding: 0.7ex;
297   }
298
299   /*a span.editme {*/
300   /*   text-decoration: underline;*/
301   /*}*/
302
303
304   table.portfoliotable {
305      width: 100%;
306
307      /*background-color: #eee;*/
```

```
308      /*background-color: #ddd;*/
309
310      /*border-top: 2px solid #eee;*/
311      /*border-left: 2px solid #eee;*/
312
313      /*border-top: 2px solid gray;*/
314      /*border-left: 1px solid gray;*/
315
316      /*border-bottom: 2px solid gray;*/
317      /*border-right: 1px solid gray;*/
318
319      empty-cells: show;
320      margin: auto;
321   }
322   table.portfoliotable tr td {
323      /*border-width: 1px;*/
324      /*border-left-style: solid;*/
325      /*border-right-style: solid;*/
326      /*border-top-style: none;*/
327      /*border-bottom-style: solid;*/
328      /*padding: 1px;*/
329      padding-left: 0.5ex;
330      padding-right: 0.5ex;
331   }
332   table.portfoliotable tr td a:hover {
333      /*color: #c22;*/
334      /*color: black;*/
335      /*color: #d4ceb4;*/
336      /*color: #cd7f32;*/
337      color: #a67d3d;      /* bronze */
338      background-color: #aaa;
339   }
340
341   /*table.portfoliotable tr.headerfooter td {*/
342   /*   border-color: #aaa;*/
343   /*   border-left-style: solid;*/
344   /*   border-right-style: solid;*/
345   /*   border-top-style: none;*/
346   /*   border-bottom-style: none;*/
347   /*}*/
348   /*table.portfoliotable tr.headerfooter td a {*/
349   /*   color: #ddd;*/
350   /*}*/
351   /*table.portfoliotable tr.headerfooter td a:hover {*/
352   /*   color: #c22;*/
353   /*   background-color: #ddd;*/
354   /*}*/
355   /*table.portfoliotable tr.headermiddle {*/
356   /**/   /*color: #eee;*/
357   /*   background-color: #9bf;*/
358   /*}*/
359
360   /*
361    * This class combo is for some reason not being
362    * triggered.  PJ Kotschy. 3Nov11.
363    */
364   table.portfoliotable tr.portfolioheader td {
365      /*border-top: 30px solid green;*/
366      /*border-bottom: 1px solid gray;*/
367      /*border-bottom: 3px solid #e92;*/
368      border-bottom: 3px solid #cd7f32;    /* This one! */
369      color: #d4ceb4;
```

```
370      /*padding-top: 1.8ex;*/
371      padding-top: 2.0ex;
372      padding-left: 1.5ex;
373   }
374   table.portfoliotable tr.portfoliofooter td {
375      border-top: 3px solid #cd7f32;
376      color: #d4ceb4;
377   }
378   table.portfoliotable tr.investmentgroup td {
379      background-color: #bbb;
380      border-bottom: 1px solid #999;
381      /*font-weight: bold;*/
382      padding-top: 0.3ex;
383      padding-bottom: 0.3ex;
384   }
385   table.portfoliotable tr.investmentheader td {
386      font-style: italic;
387      font-size: 0.85em;
388      background-color: #d4ceb4;
389      border-bottom: 1px solid gray;
390      padding-top: 0.4ex;
391      padding-bottom: 0.4ex;
392      vertical-align: bottom;
393   }
394   table.portfoliotable tr.investment td {
395      background-color: #ddd;
396      border-bottom: 1px solid #aaa;
397   }
398   table.portfoliotable tr.transaction td {
399      background-color: #eee;
400      border-bottom: 1px solid #bbb;
401   }
```

3.2.3 The `login.css` file

The visual appearance and layout of PKINVEST's login HTML screen are controlled via this file.

```
1   table.loginoutertable {
2      /*border: 2px solid #ddd;*/
3      border-top: 2px solid silver;
4      border-left: 2px solid silver;
5      border-bottom: 2px solid black;
6      border-right: 2px solid black;
7      width: auto;
8   }
9   table.logintable {
10     /*width: 40%;*/
11     width: auto;
12     /*background-color: #eee;*/
13     /*background-color: silver;*/
14     background-color: #d4ceb4;
15     empty-cells: show;
16     margin: auto;
17     border: 5px solid #333;
18  }
19  table.logintable tr td {
20     /*padding-right: 5px;*/
21     /*padding-left: 5px;*/
22     border: 1px solid #aaa;
23     padding: 3px;
24  }
```

3.2.4 The `portfolio2.m4-tmpl` file

This file is one in a set of M4-formatted files. The file expands to the HTML needed to render PKINVEST's "Portfolio" screen. Simple inspection of the content of this file reveals the presence of the abstraction layer over HTML afforded by the use of both the M7 macros and the additional macros implemented in the `NSTDIRom4/m42html.m4` file.

This file reveals not only the presence of the abstraction layer, but also the use of M7's specific `<m7dict>`, `<m7sess>` and `<m7act>` markup tags. The `<m7dict>` tag surrounds a character string key which is a reference to a value in the M7 dictionary. The tag and key are substituted for the corresponding dictionary value at that key. For example, the "`<m7dict>sess</m7dict>`" substring is substituted with the value of the login session's unique identifying key.

The `<m7sess>` tags surronds a character string key which is a reference into an M7 session dictionary. In contrast to an ordinary M7 dictionary, the state of an M7 session dictionary persists across HTTP requests between a user's browser and PKINVEST's HTTP server. For example, the "`<m7sess>port</m7sess>`" substring is substituted with the current session value of a key which uniquely identifies an entry in the `Portfolio` database table.

The `<m7act>` tag triggers an invocation of a C function implemented as part of the "Controller" component of the application. The notion of the Controller component is described in §3 on page 13. These C functions are to be considered callback functions. They have a specified function signature, and they must be registered with M7 in the "backend" source code via a call to `m7callbackSet()`. In this file, for example, the code snippet

```
<m7act>
   showPortfolioNamesInSess
   __;__
   2
   __;__
   m4beginOption(__ID__,__SELECTED__)__NAME__[[]]m4nbspace() m4endOption()
</m7act>
```

triggers the call `showPortfolioNamesInSess(m7,arg,3)`, where `m7` is an instance of the `M7` C struct, already allocated and initialised in the backend; and `arg` is an array of three character strings, initialised in the backend, at the time of the call, as

```
arg = {
   "showPortfolioNamesInSess",
   "2",
   "m4beginOption(__ID__,__SELECTED__)__NAME__[[]]m4nbspace() m4endOption()" }
```

and where, for the purpose of this example, the string `"2"` refers to the entry in the `Portfolio` database table whose unique identifier field, `id`, equals 2. It is also evident that the "`__;__`" character string is used as a delimiter in the construction of `arg`.

The substrings "`__ID__`", "`__SELECTED__`" and "`__NAME__`" found in `arg[2]` are understood by the implementation of `showPortfolioNamesInSess()`. They are appropriately substituted for the entry in the underlying `Portfolio` database table whose `id` equals 2.

The remaining M4-formatted files are structurally identical to this file and will therefore not be described further here.

```
m4_include(__INSTDIR__/m4/m42html.m4)m4_dnl
m4beginNormalPage()
m4moduleBar(portfolio)

m4beginPkinvestForm(pkinvest)

m4stateVar(req,doportfolio)
m4stateVars(sess)

m4beginActionBar()
   m4beginSelect(port)
      m4beginOption(0)[All portfolios][[]]m4nbspace() m4endOption()
      <m7act>
         showPortfolioNamesInSess
         __;__
         m4beginOption(__ID__,__SELECTED__)__NAME__[[]]m4nbspace() m4endOption()
      </m7act>
   m4endSelect()
   m4beginActionList()
      m4action(Show)m4action(Open)m4action(Edit)m4action(Add)m4action(Remove)
   m4endActionList()
m4endActionBar(Portfolio)

m4beginWorkspace()

m4beginPortfolioTable()
<m7act>
   showPortfolio3inSess
   __;__
   m4portfolioHeaderRow(12,m4_dnl
      m4_dnl m4beginHref(pkinvest?req=doportfolio&submit=Show&port=__ID__&sess=<m7dict>sess</m7
      m4_dnl    m4beginSpan(class="portfolioname")
      m4_dnl       __NAME__
      m4_dnl    m4endSpan()
      m4_dnl m4endHref()
      m4beginSpan(class="portfolioname")[[]]__NAME__[[]]m4endSpan())
   m4investmentHeaderRow()
   __INVESTMENTGROUPS__
   m4portfolioFooterRow(m4_dnl
      __SPOTVALUE__,
      __BUYSELL__,
      __RETURN__,
      __DIVIDEND__,
      __LONGGROWTHRATEPERCENT__)
   __;__
   m4investmentgroupRow(m4_dnl
      m4beginHref(pkinvest?req=doinvestmentgroup&submit=Open&group=__ID__&sess=<m7dict>sess</m7
         m4beginSpan(class="investmentgroupname")[[]]__NAME__[[]]m4endSpan()
      m4endHref(),
      __PORTFOLIOFRACTION__,
      __SPOTVALUE__,
      __BUYSELL__,
      __RETURN__,
      __DIVIDEND__,
      ,
      __LONGGROWTHRATEPERCENT__)
   __INVESTMENTS__
   __;__
   m4investmentRow(m4_dnl
      m4nbspace(),
      m4beginHref(pkinvest?req=doinvestment&submit=Open&inv=__ID__&sess=<m7dict>sess</m7dict>)
         m4beginSpan(class="investmentname")[[]]__NAME__[[]]m4endSpan()
```

```
63          m4endHref(),
64          __CODE__,
65          __PORTFOLIOFRACTION__,
66          __INVESTMENTGROUPFRACTION__,
67          __QUANTITY__,
68          __SPOTVALUE__,
69          __BUYSELL__,
70          __RETURN__,
71          __DIVIDEND__,
72          __SPOTGROWTH__,
73          __LONGGROWTHRATEPERCENT__)
74      </m7act>
75      m4endPortfolioTable()
76
77      m4endWorkspace()
78
79      m4endPkinvestForm()
80
81      m4endNormalPage()
82      m4_dnl vim: set filetype=m4 :
```

3.3 The Controller component

3.3.1 The `m7config.h` file

Many of the individual C header files encapsulate the notion of an object class. When this is the case, a single header file represents a single class, and the header file is conventionally called a *class file*. C header files conventionally have the "`.h`" file extension, and their corresponding source files have the "`.c`" file extension.

The content of a class file must contain the definition of a C struct in which the public and private data members of an object instance are stored. A convention adopted here is to denote data member privacy with an underscore character ('_') as a prefix to the corresponding field in the struct. The `M7CONFIG` class implemented here contains many public data members, two of which are `configPath` of type `PKPATH *` and `packageName` of type `char *`. But it contains no private data members.

In contrast to object data members, class data members are not represented as C struct fields, but as global file variables. For example, the `M7CONFIG` class implemented here has `MINUTESPERYEAR` as a class constant data member of type `double`. Because `MINUTESPERYEAR` is a constant, it must conventionally be assigned a contant value in this class file's corresponding source file, namely in `m7config.c`.

A class file must also contain declarations of the class's public object methods. The name of a public method must include the name of the class as a prefix, and the first argument of the public method must be a pointer to an object instance of the class. For example, in this class file, the line

```
extern int m7configInit( M7CONFIG *m7config, const char *file );
```

declares `m7configInit()` to be a public object method of the `M7CONFIG` class.

Private object methods are not declared in this manner. Instead, they are declared in a class file's corresponding source file. The declarations must be qualified as `static`, and the name of the private method must be prefixed with an underscore ('_'). For example, the `M7CONFIG` class implemented here contains no such private object methods, but the `PKINVEST` class contains `_pkinvestInvokeRequestHandler()` as a private object method which has been declared and implemented in `pkinvest.c`.

At least two public object methods must be declared and defined. These are an object constructor and an object destructor. Constructors are conventionally denoted with the substring "`Alloc`" included in the name of the method. And destructors are conventionally denoted with the substring "`Free`". For example, in this file, the lines

```
extern M7CONFIG *m7configAlloc( const char *file );
extern void m7configFree( const M7CONFIG *m7config );
```

declare `m7configAlloc()` and `m7configFree()` as a constructor and destructor for object instances of the `M7CONFIG` class.

```
#ifndef _CONFIG
#define _CONFIG

/* ----------
 * INCLUSIONS
 * ---------- */

#include <pkpath.h>
#include <m7.h>

/* -----------------
 * MACRO DEFINITIONS
 * ----------------- */

/* -----------------
 * TYPE DEFINITIONS
 * ----------------- */

/*#include <m7types.h>*/

/*typedef struct M7configs*/
typedef struct M7configs {
   PKPATH *configPath;          /* File path to the configuration file. */
   char *packageName,           /* Package name of this application. */
        *appVersion;            /* Version instance of this application. */
   PKPATH *m7dir,               /* M7 directory location. */
          *usersFile,           /* File path to the "users" file. */
          *groupsFile;          /* File path to the "groups" file. */
   PKPATH *htmlDir,             /* Directory containing the app's ".html" files. */
          *m4Dir,               /* Directory containing the app's ".m4" files. */
          *dataDir,             /* Directory containing the app's users' data files. */
          *sessionDir,          /* Directory containing the M7SESS session files. */
          *logDir;              /* Directory containing any log files. */
   int sessionLength;           /* Session length. */
   /*char *desKey;*/            /* DES encryption and decryption key.  See 'authkey.{c,h}'. */
   char *cryptSalt;             /* "Salt" character string for encryption using 'crypt()' */
                                /*    library call.  See the crypt(3) manual page.*/
   char *m4name;                /* Name of "m4" external binary. */
   PKPATH *m4path;              /* File path to the 'm4name'. */
   char *m4options;             /* Command-line options for 'm4name'. */
   char *hourFormat,            /* For an explanation of these date and time */
        *minuteFormat,          /*    representation formats, refer to the 'date(1)' */
        *dayFormat,             /*    manual page. */
        *monthFormat,
        *yearFormat;
   char *timeFormat,            /* 'timeFormat', 'monthYearFormat', 'dateFormat' */
        *monthYearFormat,       /*    and 'datetimeFormat' are variables from */
        *dateFormat,            /*    'hourFormat', 'minuteFormat', 'monthFormat' */
        *datetimeFormat;        /*    and 'yearFormat'. */
   char *integerRegexPattern,   /* Regular expression pattern for a "bounded" */
                                /*    integer number. */
        *floatRegexPattern;     /* Regular expression pattern for a "bounded" */
                                /*    floating point number . */
} M7CONFIG;

```

```
56  /* ---------------------
57   * FUNCTION DECLARATIONS
58   * --------------------- */
59
```

This declaration for the `snprintf()` function was copied from `<stdio.h>`. I needed to explicitly declare `snprintf()` because after some investigation, I discovered that by defining `_POSIX_SOURCE` with a `clang` "`-D`" command-line switch, the compilation produced the warning:

```
incompatible implicit declaration of built-in function 'snprintf'
```

```
60  /*extern int snprintf( char *, size_t, const char *, ...);*/
61
62  /*
63   * Constructors and desctructors.
64   */
65
66  extern M7CONFIG *m7configAlloc( const char *file );
67  extern void m7configFree( const M7CONFIG *m7config );
68  extern int m7configInit( M7CONFIG *m7config, const char *file );
69
70  /*
71   * Utility functions.
72   */
73
74  /* ---------------------------
75   * GLOBAL VARIABLE DEFINITIONS
76   * --------------------------- */
77
78  extern const char *LOGFNAME;
79  extern const char *CONFIGFILE;
80  /*
81   * This is terrible hack!  To have this declared like this
82   * as a global variable is too ugly for me feel at ease.
83   * But, after spending far too much time trying to find a
84   * better way for the callback functions in 'callbacks.c'
85   * to access configuration data, I decided to accept this
86   * hack.  PJ Kotschy. 22Sep11.
87   *
88   * Thinking about it again, such a global variable may not
89   * be so bad if it is thought of as a class public variable
90   * of the M7CONFIG class.  That class is declared in this
91   * header file.
92   */
93  extern M7CONFIG *GLOBALCONFIG;
94  extern const double MINUTESPERYEAR;
95  extern PKPATH DATABASEPATH[];
96
97  #endif
```

4 Acknowledgments

Thank you, Mels, for affording me time to become arguably over-preoccupied with this work, and for patiently tolerating my moments over-exuberant enthusiasm. And yes Mels, I know I should instead have been studying the physics books!

References

[1] Russell Kinnel. How expense ratios and star ratings predict success. Web site `http://www.morningstar.co.uk/uk/news/91222/p_article.aspx`.

[2] W. Vetterling W. Press, S. Teukolsky and B. Flannery. *Numerical Recipes in C.* Cambridge University Press, 2 edition, 1992.

[3] R. Johnson E. Gamma, R. Helm and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[4] Paul Kotschy. PKTECHDOC: Literate programming for non-TEX programmers, v1.0. *Still to be published*.