

Deducing the Multi-Trader Population Driving a Financial Market

Nachi Gupta* Raphael Hauser

*Oxford University Computing Laboratory, Numerical Analysis Group,
Wolfson Building, Parks Road, Oxford OX1 3QD, U.K.*

Neil F. Johnson

*Oxford University, Department of Physics,
Clarendon Building, Parks Road, Oxford OX1 3PU, U.K.*

We previously laid out a framework for predicting financial movements and pockets of predictability by deducing the heterogeneity in the multi-agent population in terms of trader types playing in an artificial financial market model [7]. This work explores extensions to this basic framework. We allow for more intelligent agents with a richer strategy set, and we no longer constrain the estimate for the heterogeneity over the agents to a probability space. We then introduce a scheme which accounts for models with a wide variety of agent types. We also discuss a mechanism for bias removal on the estimates of the relevant parameters.

Key words and phrases: Econophysics, Multi-Agent Financial Market Models, Inequality Constrained Iterative Optimization

Oxford University Computing Laboratory
Numerical Analysis Group
Wolfson Building
Parks Road
Oxford, England OX1 3QD
E-mail: nach@comlab.oxford.ac.uk

May, 2006

1 Introduction

In the current literature, there are numerous ways of making predictions over financial time-series. Some are based on stochastic differential equations modeling the expected behavior of a time-series, while others perform regressions over parameters with statistical significance. In short, there are a multitude of such approaches and techniques. Recently, there has been a movement to analyze financial time-series using a model that not only depicts similar statistical behavior at a macroscopic level but also exhibits the underlying microscopic structure of the financial market. As such, there has been a trend towards artificial market models based on multi-agent games (e.g., the minority game [4], a conceptually simple model that has shown some non-trivial behavior reminiscent of real markets).

Anyone who is attempting to model a financial market using a multi-agent trader game, with the objective of then exploiting this model to make predictions of a financial time-series, comes across two problems: (a) choosing an appropriate model and (b) estimating the heterogeneity of agents playing the game. For our example in this paper, we choose the minority game as our multi-agent game describing the market, where we allow each agent a pair of strategies, from which the agent will choose the one which performs best at each time. We continue to build upon the basic ways of estimating the composition, and hence heterogeneity, of the multi-trader population as proposed previously [7], while maintaining a measure of uncertainty for our estimate. This uncertainty is an important realization for purposes of risk analysis and identification of pockets of predictability, and thus it is very important to estimate the uncertainty intelligently based on the data, without acting overly optimistic (or pessimistic).

We also provide an extension from a probability measure (describing the heterogeneity of agents playing) to a finite measure which is not necessarily normalized to unit total weight. This has a number of benefits including a naturally stronger numerical preservation of positive definiteness in the covariance matrix for the estimates as well as the ability to make movements of unconstrained magnitude within each time step. In addition, using such a measure removes the necessity to scale the time-series, which could only be a source of further error.

We also look into the problem of estimating the probability measure over a space of agents so large that the estimation technique becomes computationally infeasible and propose a mechanism using many runs with small subsets chosen from the full space of agents to make this a more tractable problem. The final tool we provide here is for bias removal from the estimates. As a result of choosing subsets of the full agent space, an individual run can exhibit a bias in its predictions. We would like to estimate and remove this bias, for which we propose a technique that has been widely used in other Kalman Filtering applications as well.

*This author would like to thank the Clarendon Bursary for support.

2 The Artificial Market Model

For the artificial market model we choose a multi-agent game that best exhibits the microscopic dynamics that we are aware of, while maintaining the macroscopic behavior. In terms of such models, a great amount of research has been dedicated to proposing possible multi-agent games, a large class of which are Binary Agent Resource games, where a group of agents compete for a limited resource. In such games, the agents take a binary action at each time step in response to the global information available to all of the agents. And at the end of each time step, one of the actions is denoted as the winning action for which the agent would be rewarded. This winning action will also now be a part of the information set for the future. An example of a widely known binary agent resource game is the Minority Game proposed in 1997 by Challet and Zhang [4]. Over the past decade, a number of similar models and variants of the Minority Game have been proposed, and much research has been completed in this area. Throughout this paper, we present the reader with a scheme for tracking a trader population of these agents with the caveat that we trust the reader to choose the multi-agent model they find most appropriate for their specific application as we keep the tracking scheme general and adaptable to any such game.

2.1 Parameterizing the Game

We provide here one possible way of parameterizing the game to fit the methodology proposed in this paper. Note that the game need not be a Binary Agent Resource game, but for the purposes of demonstration, we assume it is. We select a time horizon window of length T over which we will score strategies for each agent. This will be a sliding window given by $(w_{k-T}, \dots, w_{k-1})$ at time step k , where $w_k = -\text{sgn}(z_k)$ represents what would have been the winning decision at time k and z_k is the difference of the price or an exchange rate series r_k .

$$z_k = r_k - r_{k-1} \tag{2.1.1}$$

Each agent may now have a set of strategies which it scores on this sliding time horizon window at each time step to choose its winning strategy which the agent will then play. Now assume we have N such agents; at each time step they each play their winning strategy which results in a global outcome for the game at that time step. As this is an artificial market model, their aggregate actions will result in an outcome which provides an idea of where we believe the time-series under analysis is moving to. If the population of strategies in use was known exactly, then one could predict the future time-series with certainty, except for a few rare time points where ties in strategies are broken randomly.

Now that we have shown how a multi-agent artificial market model might make predictions, we would like to estimate the heterogeneity of agents playing the game. For this problem, we choose to use a recursive optimization scheme similar to a Kalman Filter, but where we also force inequality constraints on the estimates so we cannot have a negative number of agents of a given type playing the game.

For example, if x_k is the vector at time step k representing the heterogeneity over N types of agents, we could write this as

$$x_k = \begin{bmatrix} x_{1,k} \\ \vdots \\ x_{N,k} \end{bmatrix} \quad (2.1.2)$$

where we force each element of the vector to be ≥ 0 .

$$x_{i,k} \geq 0, \forall i \quad (2.1.3)$$

We provided a very similar scheme previously [7], but there we further constrained our estimate to a probability space. We relax this further condition now as the benefits of lying in the probability space are outweighed by those of allowing the estimate to move out of this space. A large benefit of staying in the probability space is the ability to bound covariance matrices (as upper bounds on the probability of certain events are known). However, this benefit is mainly interesting during the start-up phase of the prediction algorithm. Over time the covariance usually tightens around the estimates where we would expect to make a prediction.

On the other hand, staying constrained to a probability space removes one degree of freedom from our system (i.e. $y_n = 1 - y_1 - \dots - y_{n-1}$). This can cause the covariance of our estimate to also become ill-formed at times and cause numerical loss of positive definiteness. In general, we artificially disallow this to happen by perturbing our matrix by a fixed amount on the diagonal elements if we see this phenomenon. However, this is a lesser problem if we choose to estimate a finite measure that is not constrained to unit total weight. In addition, we no longer worry about scaling the time-series so the possible predictions lie in a range similar to the financial movements we encounter. With a probability measure, the predicted financial movements can lie in the range $[-1, 1]$, where every agent makes the same decision. If we move away from the probability space, we no longer scale the time-series initially and the financial movements can range from $(-\infty, \infty)$, which is certainly *more* realistic.

3 Recursive Optimization Scheme

In the following subsections, we will introduce the Kalman Filter after which we will be able to discuss some desirable extensions for our applications.

3.1 Kalman Filter

A Kalman Filter is a recursive least squares implementation, which makes only one pass through the data such that it can wait for each measurement to come in real time and make an estimate for that time given all the information from the past. In addition, the Kalman Filter holds a minimal amount of information in memory at each time for a cheap computational cost in solving the optimization problem. Further, the Kalman

Filter can make a forecast n steps ahead with a covariance about this forecast, as it is a predictor-corrector system (it makes a prediction and upon observation of real data perturbs the prediction slightly as a correction, and so forth).

The Kalman Filter attempts to find the best estimate at every iteration for a system governed by the following model:

$$x_k = F_{k,k-1}x_{k-1} + u_{k,k-1}, \quad u_{k,k-1} \sim N(0, Q_{k,k-1}) \quad (3.1.1)$$

$$z_k = H_k x_k + v_k, \quad v_k \sim N(0, R_k) \quad (3.1.2)$$

Here x_k represents the true state of the underlying system, which in our case is the finite measure over the agents. $F_{k,k-1}$ is the matrix used to make the transition from state x_{k-1} to x_k . In our applications, we choose $F_{k,k-1}$ to be the identity matrix for all k since we assume the finite measure over the strategies over small regions doesn't change very drastically. And further, choosing a way to perturb the measure at each time step is a tough modeling problem. However, if desired we could incorporate any other choice of transition matrix into this model, even one that is dependent on previous outcomes. The variable z_k represents the measurement (or observation). H_k is the matrix that takes the state into measurement space. For our artificial market model, H_k will be a row vector containing the decisions based on each of the agent's winning strategies. So x_k , the measure over the agents, acts as a weighting on the decisions for each agent, and the inner product $H_k x_k$ can be thought of as a weighted average of the agents' decisions. The variables $u_{k,k-1}$ and v_k are both noise terms which are normally distributed with mean 0 and variances $Q_{k,k-1}$ and R_k , respectively.

The Kalman Filter will at every iteration make a prediction for x_k which we denote by $\hat{x}_{k|k-1}$. We use the notation $k|k-1$ since we will only use measurements provided until time step $k-1$ to make the prediction at time k . We can define the state prediction error $\tilde{x}_{k|k-1}$ as the difference between the true state and the state prediction.

$$\tilde{x}_{k|k-1} = x_k - \hat{x}_{k|k-1} \quad (3.1.3)$$

In addition, the Kalman Filter will provide a state estimate for x_k given all the measurements provided up to and including time step k . We denote these estimates by $\hat{x}_{k|k}$. We can similarly define the state estimate error by

$$\tilde{x}_{k|k} = x_k - \hat{x}_{k|k} \quad (3.1.4)$$

Since we assume $u_{k,k-1}$ is normally distributed with mean 0, we make the state prediction simply by using $F_{k,k-1}$ to make the transition. This is given by

$$\hat{x}_{k|k-1} = F_{k,k-1} \hat{x}_{k-1|k-1} \quad (3.1.5)$$

We can also calculate the associated covariance for the state prediction, which we call the covariance prediction. This is actually just the expectation of the outer product of the state prediction error with itself. This is given by

$$P_{k|k-1} = F_{k,k-1} P_{k-1|k-1} F_{k,k-1}' + Q_{k,k-1} \quad (3.1.6)$$

Notice that we use the prime notation on a matrix throughout this paper to denote the transpose. Now we can make a prediction on what we expect to see for our measurement, which we call the measurement prediction by

$$\hat{z}_{k|k-1} = H_k \hat{x}_{k|k-1} \quad (3.1.7)$$

The difference between our true measurement and our measurement prediction is called the measurement residual, which we calculate by

$$\nu_k = z_k - \hat{z}_{k|k-1} \quad (3.1.8)$$

We can also calculate the associated covariance for the measurement residual, which we call the measurement residual covariance, by

$$S_k = H_k P_{k|k-1} H_k' + R_k \quad (3.1.9)$$

Next, we will calculate the Kalman Gain, which lies at the heart of the Kalman Filter. This essentially tells us how much we prefer our new observed measurement over our state prediction. We calculate this by

$$K_k = P_{k|k-1} H_k' S_k^{-1} \quad (3.1.10)$$

Using the Kalman Gain and measurement residual, we update the state estimate. If we look carefully at the following equation, we are essentially taking a weighted sum of our state prediction with the Kalman Gain multiplied by the measurement residual. So the Kalman Gain is telling us how much to “weight in” information contained in the new measurement. We calculate the updated state estimate by

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \nu_k \quad (3.1.11)$$

Finally, we calculate the updated covariance estimate. This is actually just the expectation of the outer product of the state error estimate with itself. Here we will give the most numerically stable form of this equation, as this form prevents loss of symmetry and best preserves positive definiteness

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} (I - K_k H_k)' + K_k R_k K_k' \quad (3.1.12)$$

The covariance matrices throughout the Kalman Filter give us a way to measure the uncertainty of our state prediction, state estimate, and the measurement residual. Also, notice that the Kalman Filter is recursive, and we require an initial estimate $\hat{x}_{0|0}$ and associated covariance matrix $P_{0|0}$. Here we simply provided the equations of the Kalman Filter without derivation. For a more thorough understanding of the Kalman Filter, there are a number of available resources [2].

As we are estimating a vector in which each element has a non-negative value, we would like to force the Kalman Filter to have some inequality constraints. In the next section, we will introduce a generalization for nonlinear equality constraints followed by an extension to inequality constraints.

3.2 Nonlinear Equality Constraints

Let's add to our model given by equations (3.1.1) and (3.1.2) the following smooth nonlinear equality constraints

$$e_k(x_k) = 0 \quad (3.2.1)$$

Notice that our constraints provided in equation (2.1.3) are actually linear. We present the nonlinear case for further completeness here. We now rewrite the problem we would like to solve where we use the superscript c to denote constrained. We should also rephrase the problem we would like to solve now. We are given the last prediction and its covariance, the current measurement and its covariance, and a set of equality constraints and would like to make the current prediction and find its covariance matrix.

Let's write the problem we are solving as

$$z_k^c = h_k^c(x_k) + v_k^c, \quad v_k^c \sim N(0, R_k^c) \quad (3.2.2)$$

Here z_k^c , h_k^c , and v_k^c are all vectors, each having three distinct parts. The first part will represent the prediction for the current time step, the second part is the measurement, and the third part is the equality constraint. z_k^c effectively still represents the measurement, with the prediction treated as a "pseudo-measurement" with its associated covariance.

$$z_k^c = \begin{bmatrix} F_{k,k-1} \hat{x}_{k-1|k-1} \\ z_k \\ 0 \end{bmatrix} \quad (3.2.3)$$

The matrix h_k^c takes our state into the measurement space as before

$$h_k^c(x_k) = \begin{bmatrix} x_k \\ H_k x_k \\ e_k(x_k) \end{bmatrix} \quad (3.2.4)$$

Notice that by combining equations (3.1.3) and (3.1.4), we can rewrite the state error prediction as

$$\tilde{x}_{k|k-1} = F_{k,k-1} \tilde{x}_{k-1|k-1} + u_{k,k-1} \quad (3.2.5)$$

Now we can define v_k^c again as the noise term using equation (3.2.5).

$$v_k^c = \begin{bmatrix} -F_{k,k-1} \tilde{x}_{k-1|k-1} - u_{k,k-1} \\ v_k \\ 0 \end{bmatrix} \quad (3.2.6)$$

And v_k^c will be normally distributed with mean 0 and variance R_k^c . The diagonal elements of R_k^c represent the variance of each element of v_k^c . We define the covariance of the state estimate error at time step k as $P_{k|k}$. Notice also that R_k^c contains no off-diagonal elements.

$$R_k^c = \begin{bmatrix} F_{k,k-1}P_{k-1|k-1}F_{k,k-1}' + Q_{k,k-1} & 0 & 0 \\ 0 & R_k & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (3.2.7)$$

This method of expressing our problem can be thought of as a fusion of the state prediction and the new measurement at each iteration under the given equality constraints. Much like when we showed the Kalman Filter, we will simply write the solution here [5, 13].

$$\hat{x}_{k|k,j} = \begin{bmatrix} 0 & I \end{bmatrix} \begin{bmatrix} R_k^c & H_{k,j}^c \\ H_{k,j}^{c'} & 0 \end{bmatrix}^+ \begin{bmatrix} z_k^c - h_k^c(\hat{x}_{k|k,j-1}^c) + H_{k,j}^c \hat{x}_{k|k,j-1}^c \\ 0 \end{bmatrix} \quad (3.2.8)$$

Notice that we use the $^+$ notation on a matrix throughout this paper to denote the pseudo-inverse. In this method we are iterating over a dummy variable j within each time step until we fall within a predetermined convergence bound $|\hat{x}_{k|k,j} - \hat{x}_{k|k,j-1}| \leq c_k$ or hit a chosen number of maximum iterations. We initialize our first iteration as $\hat{x}_{k|k,0} = \hat{x}_{k-1|k-1}$ and use the final iteration as $\hat{x}_{k|k} = \hat{x}_{k|k,J}$ where J represents the final iteration.

Also, notice that we allowed the equality constraints to be nonlinear. As a result, we define $H_{k,j}^c = \frac{\partial h_k^c}{\partial x_k}(\hat{x}_{k|k,j-1}^c)$ which gives us a local approximation to the direction of h_k^c .

A stronger form for this solution can be found in past work [5, 13], where R_k^c will reflect the tightening of the covariance for the state prediction based on the new estimate at each iteration of j . We do not use this form and tighten the covariance matrix within these iterations, since in our form, we can change the number of equality constraints between iterations of j . We will find this useful in the next section. Not tightening the covariance matrix in this way is reflected in a larger covariance matrix for the estimate as well. This covariance matrix is calculated as

$$P_{k|k,j} = - \begin{bmatrix} 0 & I \end{bmatrix} \begin{bmatrix} R_k^c & H_{k,j}^c \\ H_{k,j}^{c'} & 0 \end{bmatrix}^+ \begin{bmatrix} 0 \\ I \end{bmatrix} \quad (3.2.9)$$

Notice that for faster computation times, we need only calculate $P_{k|k,j}$ for the final iteration of j . Further, if our equality constraints are in fact independent of j , we can calculate $H_{k,j}^c$ only once for each k . This would also imply the pseudo-inverse in equation (3.2.8) can be calculated only once for each k .

This method, while very different from the Kalman Filter presented earlier, provides us with an estimate $\hat{x}_{k|k}$ and a covariance matrix for the estimate $P_{k|k}$ at each time step similar to the Kalman Filter. However, this method allowed us to incorporate equality constraints.

3.3 Nonlinear Inequality Constraints

We will now extend the equality constrained problem to an inequality constrained problem. To our system given by equations (3.1.1), (3.1.2), and (3.2.1), we will also add the smooth inequality constraints given by

$$l_k(x_k) \geq 0. \quad (3.3.1)$$

Our method will be to keep a subset of the inequality constraints active at any time. An active constraint is simply a constraint that we treat as an equality constraint. An inactive constraint we will relax (ignore) when solving our optimization problem. After, solving the problem, we then check if our solution lies in the space given by the inequality constraints. If it doesn't we start from the solution in our previous iteration and move in the direction of the new solution until we hit a set of constraints. For the next iteration, this set of constraints will be the new active constraints.

We formulate the problem in the same way as before keeping equations (3.2.2), (3.2.3), (3.2.6), and (3.2.7) the same to set up the problem. However, we replace equation (3.2.4) by

$$h_k^c(x_k) = \begin{bmatrix} x_k \\ H_k x_k \\ e_k(x_k) \\ l_{k,j}^a(x_k) \end{bmatrix} \quad (3.3.2)$$

$l_{k,j}^a$ represents the set of active inequality constraints. Notice that while we keep equations (3.2.3), (3.2.6), and (3.2.7) the same, these will need to be padded by additional zeros appropriately to match the size of $l_{k,j}^a$. Now we solve the equality constrained problem consisting of the equality constraints and the active inequality constraints (which we treat as equality constraints) using equations (3.2.8) and (3.2.9). However, let's call the solution from equation (3.2.8) $\hat{x}_{k|k,j}^*$ since we have not checked if this solution lies in the inequality constrained space yet. In order to check this, we find the vector that we moved along to reach $\hat{x}_{k|k,j}^*$. This is simply

$$d = \hat{x}_{k|k,j}^* - \hat{x}_{k|k,j-1} \quad (3.3.3)$$

We now iterate through each of our inequality constraints to check if they are satisfied. If they are all satisfied, we choose $t_{\max} = 1$, and if they are not, we choose the largest value of t_{\max} such that $\hat{x}_{k|k,j-1} + t_{\max}d$ lies in the inequality constrained space. We choose our estimate to be

$$\hat{x}_{k|k,j} = \hat{x}_{k|k,j-1} + t_{\max}d \quad (3.3.4)$$

We also would like to remember the inequality constraints which are being touched in this new solution. These constraints will now become active for the next iteration and lie in $l_{k,j+1}^a$. Note that $l_{k,0}^a = l_{k-1,J}^a$, where J represents the final iteration of a given time step.

Note also that we do not perturb the error covariance matrix from equation (3.2.9) in any way. Under the assumption that our model is a well-matched model for the data, enforcing inequality constraints (as dictated by the model) should only make our estimate better. Having a slightly larger covariance matrix is better than having an overly optimistic one based on a bad choice for the perturbation [12]. Perturbing this covariance matrix correctly may be investigated in the future.

For our application, since our constraints are only to keep each element of our measure positive, we have no equality constraints and only inequality constraints. However, we needed to provide the framework to work with equality constraints before we could make the extension to inequality constraints.

In Figure 3.1, we provide a schematic diagram showing how this optimization scheme fits into the multi-agent game for making predictions.

3.4 Noise Estimation

In many applications of Kalman Filtering, the process noise $Q_{k,k-1}$ and measurement noise R_k are known. However, in our application we are not provided with this information a priori so we would like to estimate them. These can often times be difficult to approximate especially when there is a known model mismatch. We will present one possible method to approximate these [11]. We choose to match the process noise and measurement noise to the past measurement residual process.

We estimate R_k by taking a window of size W_k (which is picked in advance for statistical smoothing) and time-averaging the measurement noise covariance based on the measurement residual process and the past states. If we refer back to equation (3.1.9), we can simply calculate this by

$$\hat{R}_k = \frac{1}{W_k - 1} \sum_{j=k-W_k}^{k-1} \nu_j \nu_j' - H_j P_{j|j-1} H_j' \quad (3.4.1)$$

We can now use our choice of R_k along with our measurement residual covariance S_k to estimate $Q_{k,k-1}$. Combining equations (3.1.6) and (3.1.9) we have

$$S_k = H_k (F_{k,k-1} P_{k-1|k-1} F_{k,k-1}' + Q_{k,k-1}) H_k' + R_k \quad (3.4.2)$$

Bringing all $Q_{k,k-1}$ terms to one side leaves us with

$$H_k Q_{k,k-1} H_k' = S_k - H_k F_k P_{k-1|k-1} F_k' H_k' - R_k \quad (3.4.3)$$

And solving for $Q_{k,k-1}$ gives us

$$\hat{Q}_{k,k-1} = (H_k' H_k)^+ H_k' (S_k - H_k F_k P_{k-1|k-1} F_k' H_k' - R_k) H_k (H_k' H_k)^+ \quad (3.4.4)$$

Note that it may be desirable to keep $\hat{Q}_{k,k-1}$ diagonal if we do not believe the process noise has any cross-correlation. It is rare that one would expect a cross-correlation in the process noise. In addition, keeping the process noise diagonal has the effect of making our covariance matrix “more positive definite.” This can be done simply by setting the off diagonal terms of $\hat{Q}_{k,k-1}$ equal to 0.

It is also important to keep in mind that we are estimating covariance matrices here which must be symmetric and positive semidefinite and the diagonal elements should always be greater than or equal to zero as these are variances.

4 Estimation with Many Possible Types of Agents

It is very likely that we will come across artificial market models we would like to use that allow many different Agent Types, e.g., $N > 100$. As N grows, not only does our state space grow linearly, but our covariance space will grow quadratically. We quickly reach areas where we may no longer be in a computationally feasible region.

For example, if we look at strategies for agents playing the Minority Game and define a type of agent to have exactly 2 strategies, as we increase the memory sizes of our agents, our full set of pairs of strategies grows very quickly in relation to the memory size (e.g., $m = 1$ yields $2^{2^1} = 4$ strategies and $\binom{4}{2} = 6$ pairs of strategies, $m = 2$ yields $2^{2^2} = 16$ strategies and $\binom{16}{2} = 120$ pairs of strategies, $m = 3$ yields $2^{2^3} = 256$ strategies and $\binom{256}{2} = 32640$ pairs of strategies, $m = 4$ yields $2^{2^4} = 65536$ strategies and $\binom{65536}{2} = 2147450880$ pairs of strategies, ...). If we were interested in simultaneously allowing all possible pairs of strategies, our vectors and matrices for these computations would have dimension that would not be of reasonable complexity, especially in situations where real-time computations are needed.

In such situations, we propose selecting a subset of the full set of strategies uniformly at random and choosing these as the only set that could be in play for the time series. We can then do this a number of times and average over the predictions and their covariances. We would hope that this would cause a smoothing of the predictions and remove outlier points. In addition we might notice certain periods that are generally more predictable by doing this, which we call pockets of predictability.

4.1 Averaging over Multiple Runs

For each run j of our M runs, we have our predicted measurement at time k given by $\hat{z}_{k,j}$ and our predicted covariance for the measurement residual as $S_{k,j}$. Using the predicted measurements, we can simply average to find our best estimate of the prediction.

$$\hat{z}_k^* = \sum_{j=1}^M \frac{\hat{z}_{k,j}}{M} \quad (4.1.1)$$

And similarly, we can calculate our best estimate of the predicted covariance for the measurement residual.

$$S_k^* = \sum_{j=1}^M \frac{S_{k,j}}{M} \quad (4.1.2)$$

4.2 Bias Estimation

Since we are choosing subsets of the full strategy space, we could certainly expect that in some runs, a number of the strategies might tend to behave the same. This doesn't mean that the run is useless and provides no information. In fact, it could be the case that the run provides much information, except that the predictions always tend to be

biased in one direction or the other. So what we might like to do is remove bias from the system.

The simplest way to do this is to augment the Kalman Filter's state space with a vector of elements representing possible bias [6]. We can model this bias as lying in the state space, the measurement space, or some combination of elements of either or both. This is all done very simply. We can redefine the model for our problem as

$$x_k^b = F_{k,k-1}^b x_{k-1}^b + u_{k,k-1}, \quad u_{k,k-1} \sim N(0, Q_{k,k-1}^b) \quad (4.2.1)$$

$$z_k = H_k^b x_k^b + v_k, \quad v_k \sim N(0, R_k) \quad (4.2.2)$$

where x_k^b represents the augmented state and b_k is the bias vector at time step k

$$x_k^b = \begin{bmatrix} x_k \\ b_k \end{bmatrix} \quad (4.2.3)$$

The transition matrix must also be augmented to match the augmented state. In the top left corner, we place our original transition matrix, and in the top right corner we place $B_{k,k-1}$ representing how the estimated bias term should be added into the dynamics. In the bottom left we have the zero matrix so the bias term is not dependent on the state x_k , and in the bottom right, we have the identity matrix indicating that the bias is updated as itself exactly at each time.

$$F_{k,k-1}^b = \begin{bmatrix} F_{k,k-1} & B_{k,k-1} \\ 0 & I \end{bmatrix} \quad (4.2.4)$$

Similarly, we horizontally augment our measurement matrix, where C_k represents how the bias terms should be added into the measurement space.

$$H_k^b = \begin{bmatrix} H_k & C_k \end{bmatrix} \quad (4.2.5)$$

For the process noise, we keep the off diagonal elements as 0, assuming no cross-correlations between the state and the bias. We also generally assume no noise in the bias term and keep its noise covariance as 0, as well, but of course that can be changed easily enough if the reader would like to model the bias with some noise.

$$Q_{k,k-1}^b = \begin{bmatrix} Q_{k,k-1} & 0 \\ 0 & 0 \end{bmatrix} \quad (4.2.6)$$

We can take this bias model framework and place it into the inequality constrained filtering scheme provided earlier with model given by Equation (3.2.2), where we simply use the augmented states, when necessary, rather than the regular state space (e.g., let $x_k = x_k^b$).

5 An example on a foreign exchange rate series

We combine all of these ideas and apply these to a data set of hourly USD/YEN foreign exchange rate data from 1993 to 1994 provided by Dr. J. James of Bank One in London.

For our choice of artificial market model, we decide on using the Minority Game. We do not claim that this is necessarily a good model for the time-series, but it does have some of the characteristics we might expect to see in a real financial time-series. We look at all pairs of strategies with memory size $m = 4$ of which there are 2,147,450,880 as our set of possible types for agents. Since the size of this computation would not be tractable, we take a random subset of 5 of these types and use these 5 as the only possible types of agents to play the game, but we do 100 such runs, each time choosing 5 types at random.

In addition, we allow for a single bias removal term. We could have many more terms for the bias, but we only use 1 in order to limit the growth of the state space. And we assume the entire bias lies in a shifting of the measurements, so we choose $B_{k,k-1} = 0$ in equation (4.2.4) and C_k to be the identity matrix in equation (4.2.5) (or in our case simply 1 since C_k is 1x1).

For the analysis of how well our forecasts perform, we calculate the residual log returns and plot these. Given our time-series, we can calculate the log return of the exchange rate as $l_k = \log(r_k) - \log(r_{k-1})$. Note that based on our definition for z_k from equation (2.1.1), we can write the log return also as $l_k = \log(r_{k-1} + z_k) - \log(r_{k-1})$. Similarly, we can define our predicted forecast for the log return as $\hat{l}_k = \log(r_{k-1} + \hat{z}_k) - \log(r_{k-1})$. Given these two quantities, we can calculate the residual of the predicted log return and the observed log return as $\tilde{l}_k = l_k - \hat{l}_k$. Using the delta method [3], we can also calculate the variance of this residual to be $\frac{S_k}{(r_{k-1})^2}$.

We decide to do 100 such runs, which we average over using the method described in Section 4.1. A good test for whether our variances are overly optimistic is to check if the measure satisfies the Chebyshev Inequality. For example, we can check visually that no more than about $\frac{1}{9}$ of the residuals lie within 3σ 's. This must be true of any probability measure. We plot the residual log return along with 3σ bounds and the standard error of the mean in Figure 5.1. Visually, we would say the residuals in Figure 5.1 would certainly satisfy this condition if they were centered about 0 (using further bias removal should do this).

6 Conclusion

We have presented here some advanced techniques which extend the numerical methods provided in our previous report [7]. When coupled with a more appropriate underlying market model, these techniques could provide useful insight into the composition of the underlying multi-trader population, and hence yield superior prediction estimates of the resulting financial time-series which is generated by this population.

References

- [1] Jørgen Vitting Andersen and Didier Sornette, 2005. A mechanism for pockets of predictability in complex adaptive systems. *Europhysics Letters*, **70**(5): 697–703.

- [2] Yaakov Bar-Shalom, X. Rong Li and Thiagalingam Kirubarajan, 2001. *Estimation with Applications to Tracking and Navigation*. John Wiley and Sons, Inc.
- [3] George Casella and Roger L. Berger, 2002. *Statistical Inference*. Thomson Learning.
- [4] Damien Challet, Matteo Marsili and Yi-Cheng Zhang, 2005. *Minority Games*. Oxford University Press.
- [5] Y. T. Chiang, L. S. Wang, F. R. Chang and H. M. Peng, October 2002. Constrained filtering method for attitude determination using gps and gyro. *IEE Proceedings - Radar, Sonar, and Navigation*, **149**(5): 258–264.
- [6] Bernard Friedland, August 1969. Treatment of bias in recursive filtering. *IEEE Transactions on Automatic Control*, **14**(4): 359–367.
- [7] Nachi Gupta, Raphael Hauser and Neil F. Johnson, June 2005. Using artificial market models to forecast financial time-series. In *Workshop on Economic Heterogeneous Interacting Agents 2005*, e-print physics/0506134 at xxx.lanl.gov.
- [8] Paul Jefferies and Neil F. Johnson, July 2002. Designing agent-based market models. e-print cond-mat/0207523 at xxx.lanl.gov.
- [9] Neil F. Johnson, Paul Jefferies and Pak Ming Hui, 2003. *Financial Market Complexity*. Oxford University Press.
- [10] Neil F. Johnson, David Lamper, Paul Jefferies, Michael L. Hart and Sam D. Howison, October 2001. Application of multi-agent games to the prediction of financial time-series. *Physica A: Statistical Mechanics and its Applications*, **299**(1–2): 222–227.
- [11] Peter S. Maybeck, 1982. *Stochastic Models, Estimation and Control*, Volume 2. Academic Press, Inc.
- [12] Dan Simon and Donald L. Simon, February 2003. Kalman filtering with inequality constraints for turbofan engine health estimation. Technical Report A491414, National Aeronautics and Space Administration, John H. Glenn Research Center at Lewis Field.
- [13] L. S. Wang, Y. T. Chiang and F. R. Chang, November 2002. Filtering method for nonlinear systems with constraints. *IEE Proceedings - Control Theory and Applications*, **149**(6): 525–531.

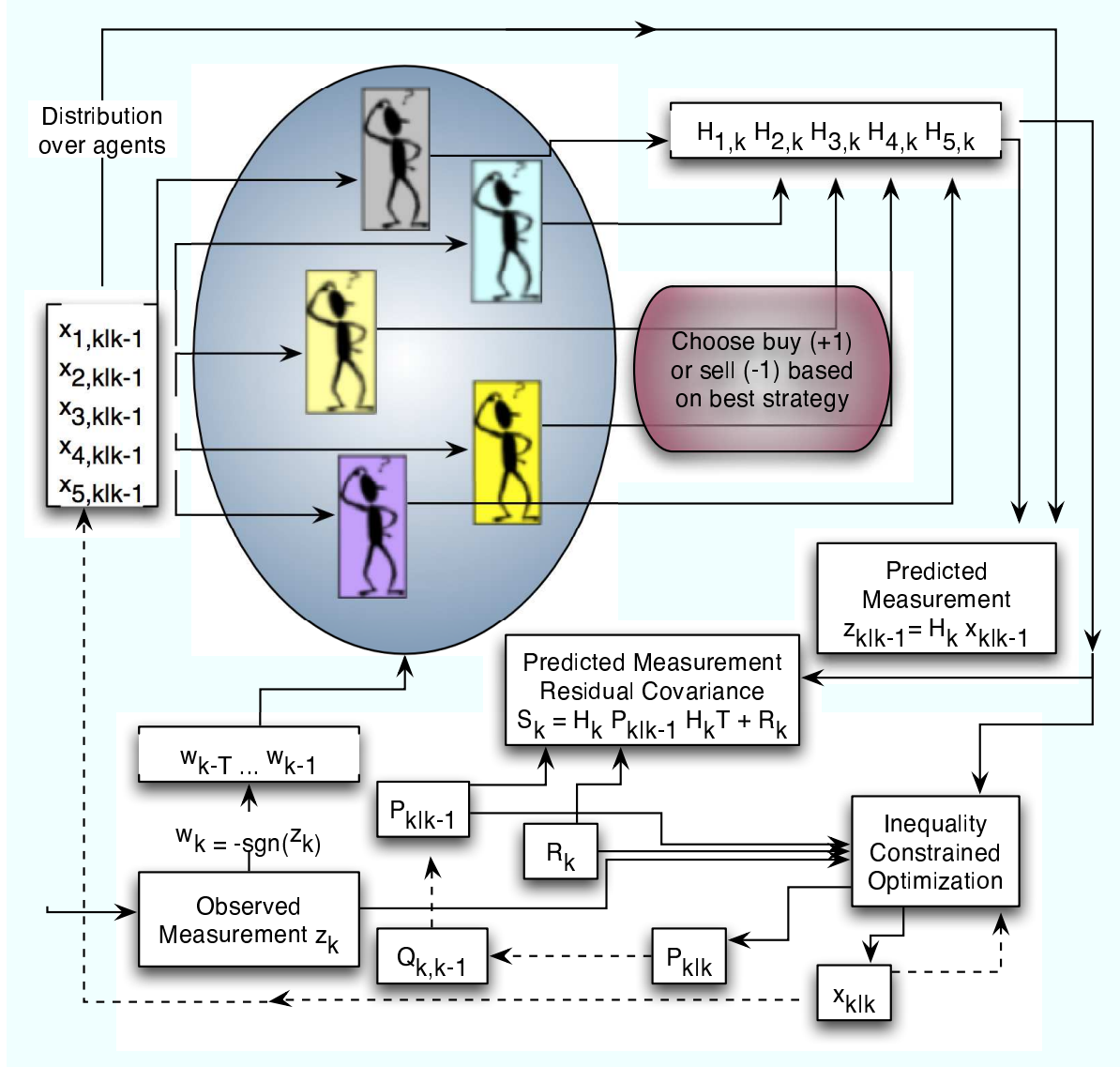


Figure 3.1: We summarize the main idea of the recursive method we describe for predicting the heterogeneity over agents playing (where we drop the $\hat{\cdot}$ notation). Here we describe a situation with 5 types of agents, where each type has more than one strategy. They each score their strategies over the sliding time horizon window ($w_{k-T} \dots w_{k-i}$) and choose the best one. H_k then represents the decisions they each make in this time step, which in the case of a Binary Agent Resource Game is +1 or -1. Taking the dot product of the frequencies over the agents and their decisions, we arrive at our prediction for the measurement. We then allow the recursion into the optimization technique. Notice that since we choose $F_{k,k-1}$ as the identity matrix for all k , we have simply omitted it in this diagram, and we also assume initial conditions are provided. In the next subsection, we describe how we arrive at the noise parameters $Q_{k,k-1}$ and R_k , which appear in the diagram.

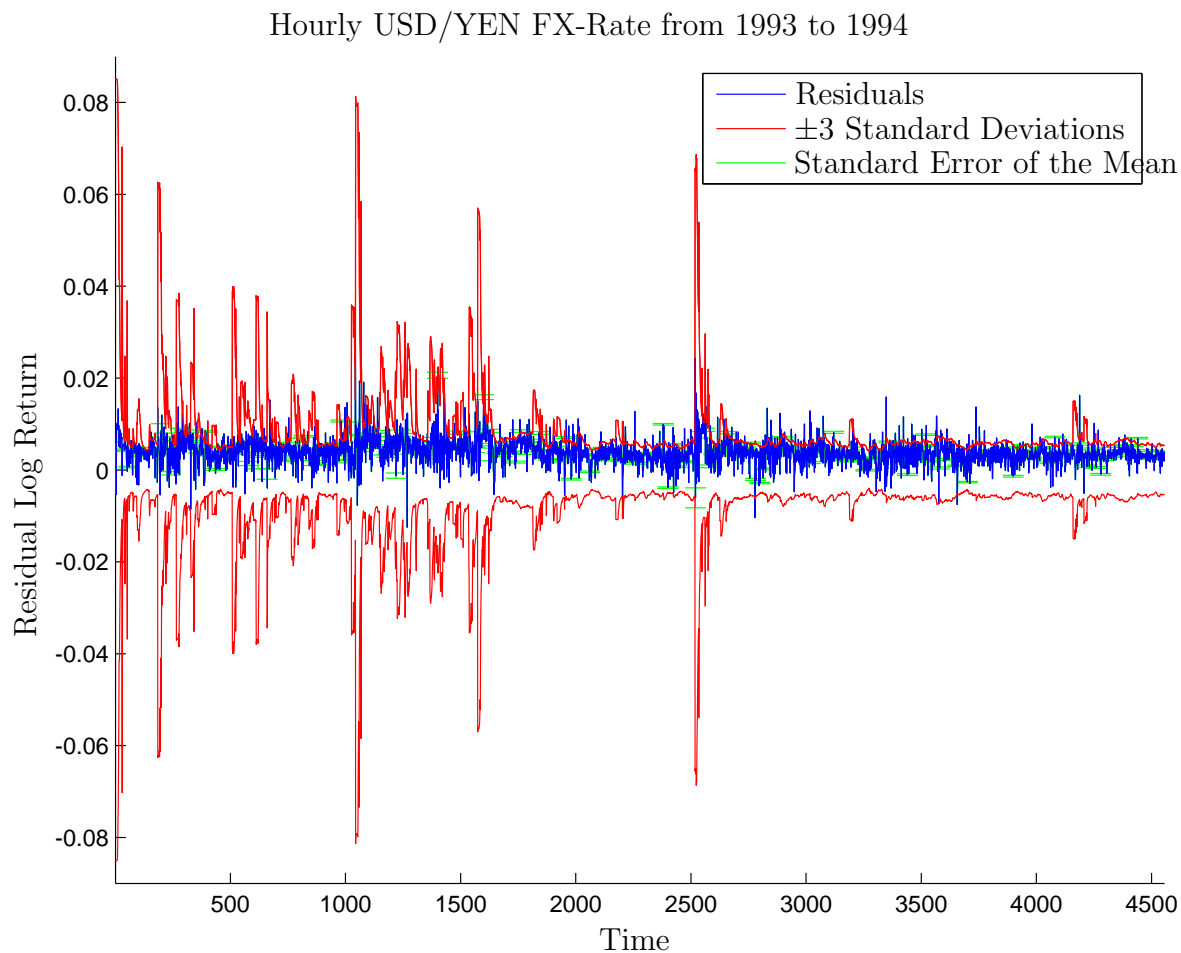


Figure 5.1: We show the residuals of the log returns plotted with 3σ 's centered about 0 and the standard error of the mean over the 100 runs. Despite, the one parameter bias removal, we still see a general bias in the data, without which the residuals look much cleaner. Perhaps modeling the bias in higher dimension would remove this.