

# Time Warp Simulation of Timed Petri Nets: Sensitivity of Adaptive Methods \*

Alois Ferscha and Michael Richter  
Institut für Angewandte Informatik  
Universität Wien  
Lenaugasse 2/8, A-1080 Vienna, AUSTRIA

## Abstract

*The unthrottled optimism underlying the Time Warp (TW) parallel simulation protocol can lead to excessive aggressiveness in memory consumption due to saving state histories, and waste of CPU cycles due to overoptimistically progressing simulations that eventually have to be “rolled back”. Furthermore, in TW simulations executing in distributed memory environments, the communication overhead induced by the rollback mechanism can cause pathological overall simulation performance. In this work direct optimism control mechanisms are used to overcome these shortcomings by probabilistically controlling simulation progression based on the forecasted time stamp of forthcoming messages. Several forecast methods are presented and their performance is compared for very large Petri net simulation models executed with the TW protocol on the Meiko CS-2.*

## 1 Introduction

For the quantitative analysis of timed Petri net models representing realistically sized physical time dynamic systems, traditional evaluation techniques like analysis or sequential discrete event simulation tend to become practically intractable. To be able to cope with very large models, parallel (distributed) simulation mechanisms are demanded and parallel simulation software tools are necessary for the massively parallel execution of such models.

A variety of approaches has been followed in the literature to adopt standard parallel and distributed simulation techniques to the concurrent execution of timed Petri nets [9]. *Parallel* simulation strategies either exploit algebraic properties of timed transition firing semantics [1], or the synchronism of time progression in Petri nets with discrete timing, yielding massively parallel simulators dedicated to SIMD execution. *Distributed* simulation approaches involve the spatial decomposition of Petri nets and the asynchronous parallel execution of submodels by

so called *logical processes* (LPs). LP simulations of Petri nets have been developed along conservative Chandy/Misra/Bryant (CMB), as well as along the optimistic Time Warp (TW) intra-LP causality preservation protocols [11]. CMB protocols execute in each LP events that occur in the respective submodel in nondecreasing order of their occurrence timestamp, while strictly preventing from the possibility of any event causality violation across LPs. Such protocols have been adopted for the concurrent execution of Petri nets in [19, 17, 5, 6]. The primary motivation for TW protocols is that events which occur in different LPs – irrespective of their occurrence time stamp – might not affect one another, thus giving rise to their parallel, out-of-timestamp-order execution. This view has led to TW based Petri net simulators [5].

While letting causality errors occur, TW employs a *rollback* mechanism to recover from causality violations immediately upon or after their detection. The rollback procedure in turn relies on the reconstructability of past states which can be guaranteed by a systematic state saving policy and corresponding state reconstruction procedures, causing respective overheads. More than that can TW suffer from communication overheads to a threatening extent: In situations where event occurrences are highly dispersed in space and time, rollback invocations can recursively involve long cascades of LPs which will eventually terminate. An excessive amount of local and remote state restoration computations is the consequence of the annihilation of effects that have been diffused widely in space and too far ahead in simulated time, consuming considerable amounts of computational, memory and communication resources while not contributing to the simulation as such. This pathological behavior is basically due to the “unlimited” optimism assumption underlying TW, and has often been referred to as *rollback thrashing*. Techniques to adaptively, i.e. according to the parallelism inherent in the simulation model, avoid these overheads are the matter of investigation in this work.

**Outline** This paper, after presenting related work on optimism control in TW (Section 2), recalls the adap-

\* This work was supported by the Oesterreichische Nationalbank under grant No. 5069, and the Human Capital and Mobility program of the EU under grant CHRX-CT94-0452 (MATCH).

```

Simulation_Engine(i) {
s1 iel=initial(simulator, model, forecast method);
  while(ie=next_ie(iel)) chronological_insert(ie,EVL);
  log_new_state();
s2 while (GVT < ENDTIME || GVT_term) {
s3   while (m=read_next_input_buffer_message()) {
      #if THROTTLED
s4     if (positive(m)) update_statistics(m);
      #endif
s5     if ( ts(m) < LVT) process_Straggler(m);
s6     if (!remove_dual(m,IQ)) insert(m,IQ);
    }
    #if THROTTLED
s7     estimate(LVTW, confidence);
s8     if (get_time_first_EVL_or_IQ() ≥ LVTW)
s9       if (random() < confidence)
          delay(AVE_EVENT_PROC_TIME);continue;
    #endif
s10  advance_GVT(); fossil_collection();
s11  if (memory_used > MEMORY_LIMIT) continue;
s12  e = get_first_EVL_or_IQ();
      if (e) process_event(e);
s13  fill_OB(OQ);
      send_out_contents(OB);
    }
s14 analyse_stack(); clean_up();
}

```

Figure 1: Throttled Time Warp Simulation Engine

tive throttling strategy based on statistical arrival process analysis and message time stamp prediction. In Section 3 a collection of such analysis methods is developed at different levels of sophistication and with increasing computational complexity. In a case study in Section 4, the performance sensitivity of these forecast methods is evaluated with a very large Petri net business process model. Conclusions are drawn in Section 5.

## 2 Adaptive Optimism Control in TW

Several approaches have been followed in the literature to limit the optimism of TW. The Adaptive Time Warp (ATW) protocol [2] allows the execution of events  $e$  with time stamp  $t(e)$  only if  $t(e) \in [t, t + \Delta)$ , where  $\Delta$  is adapted dynamically according to the number of  $lc$  constraint violations in the past. In [12] a protocol is proposed that temporarily blocks an LP if a potential straggler messages (that would induce rollback) is anticipated. The Local Adaptive Protocol (LAP) [15] in much the same way uses statistical information from observing the simulation performance on the fly to dynamically adjust optimism control parameters. An explicit cost function to determine whether it is more “cost effective” to execute an arriving message instantaneously despite rollback hazard, or to delay its execution to avoid potential

```

process_Straggler(m) {
r1 dual=dual_exists(m,IQ);
r2 if ((positive(m)&& !dual) || (negative(m)&& dual))
  {
r3   LVT=restore_earliest_state_before(ts(m));
r4   ant_evl=generate_antimessages(LVT);
      while(ee=next_ee(ant_evl))
        chronological_insert(ee,OQ);
  }
}

process_event(e) {
e1 new_ev, pre_evl, ext_evl =modified_by_e(e);
  while(ie=next_internal_e(new_evl))
e2   chronological_insert(ie,EVL);
  while(ie=next_preempted_int_e(pre_evl))
e3   remove_event(ie, EVL);
e4 LVT = ts(e);log_new_state();
  while(ee=next_external_e(ext_evl))
e5   if (!dual_update(ee,OQ))
      chronological_insert(ee,OQ);
}

```

Figure 2: Rollback Procedure and Event Processing

rollback/antimessages is reported in [13]. A combination of controlling optimism, and an automatic adjustment of the amount of memory required in TW is approached in the Adaptive Memory Management (AMM) scheme [7]. All these protocols use statistical data that reflects the central tendency (e.g. average increase of local virtual time and average (real and simulated) message interarrival time in LAP; average commitment rate in AMM) to determine parameters for optimism control in TW. However, optimism control based on averages is not promising when simulation models incur “phases” of different LP synchronization behavior, where reactivity and an automatic readjustment of dynamically evolving changes in the simulation workload control parameters from “phase” to “phase” is desired.

In [10], we have developed an LP synchronization protocol which throttles the TW optimism probabilistically. (A sketch of the protocol is given in Figure 1, with conditional compile directives for the throttled version of the TW protocol.) The adaptive scheme, as opposed to CMB which would block the CPU if there is the chance of a message with a time stamp in the past (straggler message) arriving at the LP, and opposed to TW which would optimistically progress LVT even if the chance for receiving a straggler message is very high, executes scheduled process elements based on the probability of not receiving a straggler (see [10] for detailed presentation of the adaptive protocol). This approach reduces the waste of CPU cycles due to blocking in CMB, while at the same time

reducing the communication cost caused by annihilation messages in case of rollback in TW. Technically, the degree of “optimism” underlying TW is regulated according to hypotheses that LPs are establishing during simulation on the amount of available model parallelism as observed from the time stamps carried by arriving messages. With the help of a forecast for the forthcoming message, the “aggressiveness” of TW is adaptively throttled to that point in the spectrum between unlimited optimism and extreme pessimism, that is the best compromise for the a particular Petri net model.

Specifically the adaptive TW protocol operates as follows: Assume the time stamp of the forthcoming message  $m_{i+1}$  to arrive at LP<sub>*j*</sub> to be  $t(m_{i+1})$ , and the current instant of LVT to be  $LVT_j$ . TW with unthrottled optimism would execute any event  $e$  scheduled for occurrence, irrespective of its occurrence time  $ot(e)$ . Assuming  $t(m_{i+1})$  is equally likely in  $[s, t]$ , TW will have to enforce rollback for any  $e$  with  $ot(e) > t$ . Clearly, the induced rollback(s) and communication overhead for sending annihilation messages could have been avoided. In the adaptive protocol, the time stamps of arriving messages ( $t(m_{n-i-1}), t(m_{n-i}), \dots, t(m_i)$ ) are statistically analyzed to forecast  $t(m_{i+1})$  as  $\hat{t}(m_{i+1})$ . Assuming the confidence in the forecast to be  $(1 - \alpha)$ , then an event  $e$  scheduled with  $ot(e)$  is executed with probability

$$\mathcal{P}[\text{execute } e] = 1 - \frac{1}{1 + e^{-\left(\frac{LVT_j - \hat{t}(m_{i+1})}{\alpha(1-\alpha)100}\right)}},$$

otherwise its execution is delayed (the CPU is blocked) for  $\epsilon$  time units where  $\epsilon$  is the average CPU time required to simulate one process element execution. (Note that the higher the confidence level  $(1 - \alpha)$ , the steeper is the ascent of the delay probability as LVT progresses towards  $\hat{t}$ .) After LVT progression has surpassed the estimate  $\hat{t}$ , delays become more and more probable, expressing the increasing rollback hazard the LP runs into. A general observation is that with  $(1 - \alpha) \approx 1$ , throttling yields a synchronization behavior close to CMB, whereas with  $(1 - \alpha) \approx 0$ , optimism is as unlimited as in TW [10].

A critical issue with the adaptive protocol is the way in which  $\hat{t}(m_{i+1})$  is computed: forecasts based on the central moment of the arrival history will cause less computational overhead, but may generate inadequate predictions, whereas methods based on a time series analysis giving better predictions can become excessive in memory and CPU cycle consumption.

Through the rest of this paper we shall investigate a variety of methods for computing  $\hat{t}(m_{i+1}) = t(m_i) + \hat{\Delta}(\delta_1, \delta_2, \dots, \delta_n)$  from the observed token time stamp differences  $\delta_k = t(m_{i-n+k}) - t(m_{i-n+k-1})$

### 3 Predicting Time Stamps

It is obvious that the overall performance of adaptive throttling in TW with the approach presented above is reliant on the quality of the predictor  $\hat{t}$  and the confidence  $\alpha = \zeta(\hat{t})$  in the predictor. This quality is in turn influenced by the amount of information available for forecasting, i.e. the size of the observation window  $n$  maintained for each input channel in every LP, as well as the choice of the forecast procedure. Generally, the larger  $n$ , the more information on the arrival history is available in the statistical sense. Considering much of the arrival history will at least theoretically give a higher prediction precision but will also consume more memory space. Intuitively, complex forecast methods could give “better” predictions than trivial ones, but are liable to intrude on the simulation engine with an unacceptable amount of computational resource consumption. Therefore, *incremental* forecast methods of low memory complexity are recommended, i.e. procedures where  $\hat{t}(m_{i+2})$  can be computed from the previous forecast  $\hat{t}(m_{i+1})$  and the actual observation  $t(m_{i+1})$  in  $\mathcal{O}(c)$  instead of  $\mathcal{O}(cn)$  time.

#### 3.1 Central Tendency Forecasts

##### 3.1.1 Arithmetic Mean based Forecasting

A trivial forecast for  $\hat{t}(m_{i+1})$  is the one that uses the arithmetic mean of time increments  $\hat{\Delta} = \bar{\delta}$ , with  $\zeta(\hat{t}) = 1 - \frac{s}{\bar{\delta}}$ , where  $s$  is the empirical standard deviation of  $\delta_1, \delta_2, \dots, \delta_n$ . Both  $\bar{\delta}$  and  $s$  can be computed incrementally, i.e. given  $\bar{\delta}_n$  and  $s_n$  after  $n$  messages. Upon a new arrival carrying a time increment  $\delta_n$ , the new  $\bar{\delta}_{n+1}$  and  $s_{n+1}$  are computed as

$$\bar{\delta}_{n+1} = (n\bar{\delta}_n + \delta_{n+1}) / (n + 1). \quad (1)$$

$$\begin{aligned} s_n^2 &\stackrel{\text{def}}{=} \frac{1}{n-1} \sum_{i=1}^n (\delta_i - \bar{\delta}_n)^2 = \frac{n(\bar{\delta}_n^2 - \bar{\delta}_n^2)}{n-1} \quad (2) \\ \Rightarrow s_{n+1}^2 &= \frac{n+1}{n} (\bar{\delta}_{n+1}^2 - \bar{\delta}_{n+1}^2) \quad (3) \end{aligned}$$

The main advantage of using the arithmetic mean as a forecast is its incremental computation involving  $\mathcal{O}(1)$  operations. (Note that for the incremental computation of  $s^2$  also  $\bar{\delta}^2$  needs to be computed incrementally.) A potential disadvantage appears when the distribution of  $\delta_i$  is skewed –the prediction would then either consistently over- or underestimate the next token time. In this case a forecast based on the median appears more appropriate.

##### 3.1.2 Median based Forecasting

Median computation involves sorting the vector of observed message time increments. The forthcoming

message time is predicted to be  $\hat{t} = t_n + \hat{\Delta}$ , where  $\hat{\Delta}$  is the median of the time increments. Let  $X$  contain the sorted vector  $(\delta_1, \delta_2, \dots, \delta_n)$ , the median (or 50% quantile  $q_{.50}$ ) is defined as:

$$Me(X) = \begin{cases} X_{[\frac{n+1}{2}]}, & \text{if } n = \text{odd} \\ \frac{1}{2}(X_{[\frac{n}{2}]} + X_{[\frac{n}{2}+1]}), & \text{if } n = \text{even} \end{cases} \quad (4)$$

The primary advantage of the median is its property of the smallest possible deviation:

$$\sum_{i=1}^n |X[i] - Me(X)| \leq \sum_{i=1}^n |X[i] - c| \quad \forall c \in \mathbb{R},$$

and its robustness against outliers in the arrival stream. Instead of the standard deviation, the *interquartile range* ( $q_{.70} - q_{.25}$ ) could also be used to determine the confidence. The disadvantage of the median is that it prohibits an incremental computation, causing an  $O(\log(n))$  sorting effort in each step.

### 3.1.3 Exponential Smoothing

*Exponential smoothing* is a method to remove seasonal trends from time series and allows for a weighting of the more recent history over the less recent history or vice versa. Here  $\hat{\Delta}$  is a weighted moving average of  $\delta_k, \delta_{k-1}, \dots$ , with weights decreasing exponentially on the weighting factor  $\alpha \in (0, 1)$ , incrementally computed as:

$$\hat{\Delta}_{k+1} = \alpha \delta_k + (1 - \alpha) \hat{\Delta}_k \quad (5)$$

To approximate the arrival process as “smooth” as

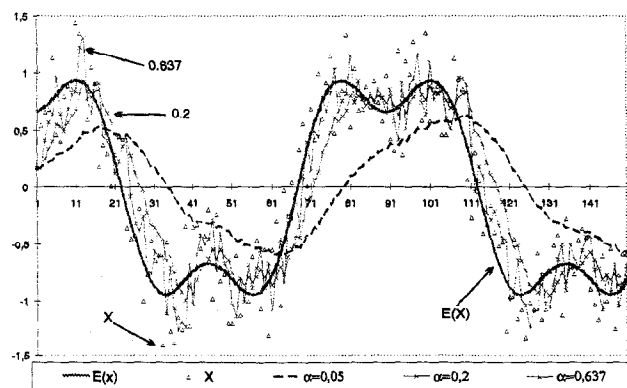


Figure 3: Effect of Exponential Smoothing.

possible,  $\alpha$  needs to be chosen so as to minimize the residuals:

$$\min \sum |\delta_k - \Delta_k(\alpha)| \quad 0 \leq \alpha \leq 1$$

Practically this minimum is approximated by computing the sum of residuals for  $q * 10$  different values of  $\alpha$  within the interval  $[0, 1]$  after every  $n^{th}$  incoming message.

Figure 3 shows a sample arrival process  $X$  and the effect of varying smoothing parameters  $\alpha = 0.05$ ,  $\alpha = 0.2$ , and  $\alpha = 0.637$ . In the particular example, the parameter  $\alpha = 0.637$  minimizes the residuals exposed by  $X$ .

## 3.2 Time Series Forecasting

In cases where the arrival process  $(\delta_1, \delta_2, \dots, \delta_n)$  exhibits trends and seasonal behavior, the previous forecast methods are prone to pathological behavior due to repeated over- and underestimation of the forthcoming message time stamp. To cope with arrivals that exhibit phases and/or trends, the identification the time series underlying the arrival process is important. Forecasts based on such time series models will involve significantly higher computational complexity, but will deliver more robust forecasts. The technical details for describing the time stamp increment process as an unknown stochastic process, i.e. considering  $(\delta_1, \delta_2, \dots, \delta_n)$  as realizations of random variables  $X_t, t \in T$  are given in Appendix A.

## 3.3 Kalmanfilter Forecasting

Yet another approach for estimating the time stamps of forthcoming messages is to use a *Kalmanfilter model*, which (as an improvement) supports a recursive forecast procedure (Appendix B). Still both, the forecast methods based on time series analysis and the Kalmanfilter approach suffer from the very complex computation, which can potentially overwhelm the gain of adaptive rollback prevention. On the other hand can both approaches provably recognize correlated arrivals and achieve a better prediction precision than methods based on central moments. In the following case study we shall evaluate the potentials of all of these methods as applied to the distributed TW simulation of very large timed Petri net models.

## 4 Case Study

### 4.1 Sensitivity of Adaptive Methods

To investigate the performance sensitivity of adaptive methods, timed Petri net models describing the document flow in a very large business organization were chosen (more than 1 M documents, more than 60 offices). To avoid presenting the full complexity of the models in this paper, we just present a simplified abstraction as shown in Figure 4. In this abstraction, the documents flowing into a work area (office) are represented by tokens arriving at a place  $P_{in}$ . A transition  $T_{ext}$  propagates documents to other offices (regions), or routes them to the appropriate places of execution within that work area, i.e. region. In the experiments we study an office system consisting of 64 offices, connected to each other by the arcs originating from respective  $T'_{ext}$ s, and discharging into

$P_{in}$  places. I.e., the execution of  $T_{ext}$  in region  $R_i$  causes a document (token in the Petri net model) to be transferred to the successor region  $R_j$ . If  $R_i$  and  $R_j$  are assigned to different LPs assigned to different processors, the TW protocol generates and sends a message for each document flowing from  $R_i$  to  $R_j$ , whereas documents flowing within one region are local and do not invoke any message sending. See [14] for a more detailed description of the kind of models investigated. To introduce sufficient variation with respect

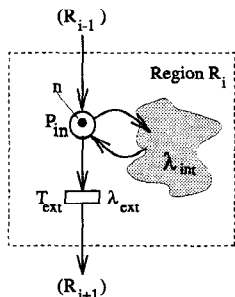


Figure 4: Petri net region of a Document Flow System

to model behaviors, we examine transitions  $T$  adhering to the race firing policy and three different kinds of enabling time  $\tau$ : (i) exponential ( $\tau(T_i) \sim \exp(\lambda_i)$ ), (ii) deterministic ( $\tau(T_i) = (1/\lambda_i)$ ) and (iii) mixed ( $\tau(T_i) \sim (4/\lambda_i + \exp(\lambda_i))$ ).

For the analysis of the TW sensitivity using the above model, the following parameters were considered:

- **Number of Available Processors:** With an increasing number of processors, more and more inherent model parallelism may be exploited but interprocessor communication costs also rise. At some point communication costs are expected to dominate over useful simulation work.
- **Initial Token Count:** The larger the number of tokens in one region (i.e. initial tokens in  $P_{in}$ ), the larger the model parallelism, i.e. the number of possible concurrent process element executions. However, as the number of tokens in the region increases, so does also the number of events which must be managed by the simulation engine.
- **Token Flow Rate among Regions:** By adjusting the ratio of the firing rates  $\lambda_{int} : \lambda_{ext}$ , it is possible to simulate the behavior of regions with varying degrees of “locality”, i.e. a large  $\lambda_{int} : \lambda_{ext}$  ratio simulates a model where the majority of the simulation steps (i.e. process execution) affect only processes within the LP’s own partition whereas a small ratio causes increased communication overheads. We shall refer to the

value of  $\lambda_{int} : \lambda_{ext}$  as the internal/external event ratio.

- **Forecast Method:** We use all the forecast methods to estimate the arrival time of the next message as described above.
- **Throttling Delay:** We consider three cases to delay the execution of a scheduled event. In the first case, once the simulator decides not to execute the next event, control just loops back to the reading the input buffers section, thus implicitly delaying its execution. In the two other cases the simulation is explicitly delayed for a certain amount of time: the delay is either set to the amount of CPU time used to execute one event on average, or the amount of CPU time used to execute one event at maximum.

All the experiments reported in the rest of this paper were obtained from a significant number of execution runs on a 134 node Meiko CS-2.

## 4.2 TW with implicit Throttling

In the first set of experiments, the performance of TW with implicit throttling delays was investigated for  $N = 2^0, \dots, N^5$  processors of the CS-2, each simulating  $64/N$  regions of the kind in Figure 4. All the transitions in all the regions had exponential enabling delays. As a performance measure the classical “event rate” was translated into a metric called the “acceleration factor”, which stands for the total number of committed transition firings per processor per unit time (transition firings that do not contribute useful simulation work, i.e. are subject to annihilation in a rollback situation are not counted). Figure 5 compares the acceleration factor for an internal/external event ratio of 1 to 10000. It is seen that for a large number of initial tokens the simulation engines using adaptive methods are able to execute 15 % more transition firings (events) than a simulation engine using plain TW, given a sufficiently large token population. In cases with a low number of initial tokens, however, plain TW performs better than adaptive TW. Almost irrespective of the simulation protocol, if there are less initial tokens, simulation engines have less simulation work: the average length of the event list EVL is shorter, and therefore state saving and possible rollbacks are performed faster. In our model with 32 processors the average event processing and state saving time is  $3981\mu\text{sec}$  for 1024 documents (tokens) and  $2872\mu\text{sec}$  for 64 documents. The average rollback time decreases from  $105153\mu\text{sec}$  to  $1024\mu\text{sec}$ .

For an analysis of the gain of the adaptive mechanisms we express the relation between the costs for rollbacks and the forecast method as

$$\sum_{n=0}^N t_{n, \text{Rollback}(TW)} \geq \sum_{n'=0}^{N'} t'_{n', \text{Rollback}(Method)} +$$

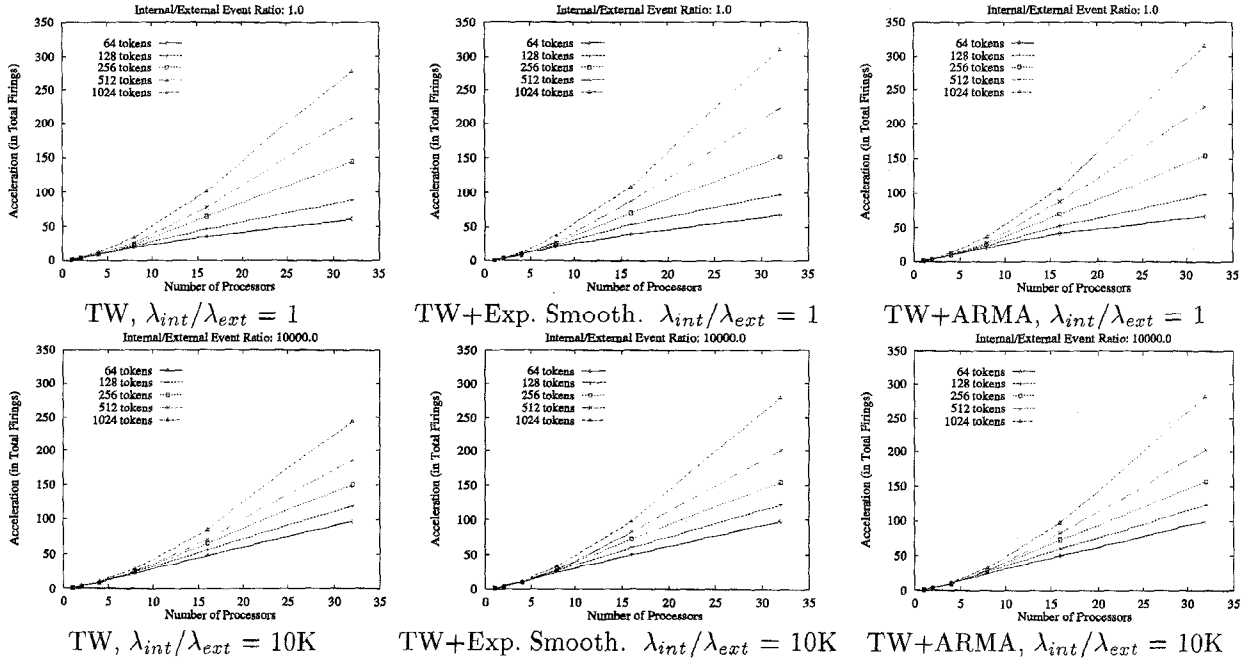


Figure 5: Performance Sensitivity with Respect to Internal/External Event Ratio

$$+ \sum_{m=0}^M t_{m,Update} + \sum_{i=0}^I t_{i,EstimateMethod}, \quad (6)$$

where  $N$  is the number of rollbacks in plain TW,  $t_{n,Rollback(TW)}$  is the time for performing the  $n^{th}$  rollback,  $N'$  and  $t'_{n',Rollback(Method)}$  are the parameters for TW with a forecast method.  $M$  is the number of external positive messages and  $I$  is the number of estimate function calls. Under normal conditions we can say, that  $M > N$  and  $I \gg M$ . The adaptive methods perform better (only) if the aggregated times on the right side of equation (6) are smaller than the sum of the rollback times in plain TW. In our examples this seems to be the case at a “token load” of 512 and higher. Otherwise the computational forecast overhead dominates the gain.

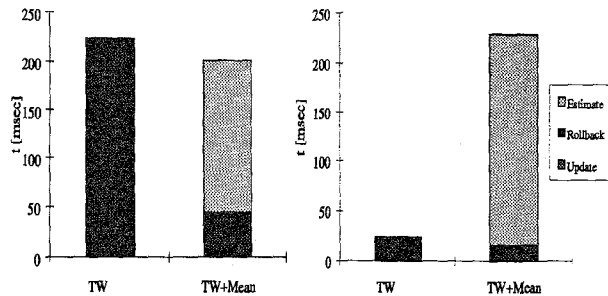


Figure 6: Token Load: 1024 (left), 64 (right)

Figure 6 shows a comparison of the aggregated costs

of rollbacks and calls of the forecast function for subnets with an internal/external event ratio of 1 and 1024 tokens (left) and 64 tokens (right), mapped on 32 processors, using plain TW and TW with arithmetic mean forecast. Both cases reveal a reduction of the rollback cost. The additional calls of the forecast function in the loaded system are overwhelmed by the gain with respect to rollbacks. In the case with 64 initial tokens, however, the CPU time of the estimate function calls is much larger than the rollback cost reduction.

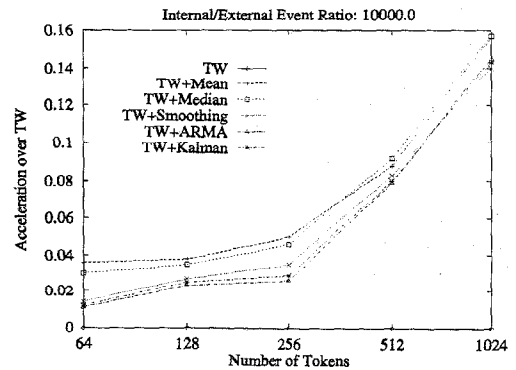


Figure 7: Adaptive Throttling Performance Gain

A comparison of the relative accelerations as induced by the various forecast strategies (for the particular case of  $\lambda_{int}/\lambda_{ext} = 10K$  executing on 32 nodes of the CS-2) is summarized in Figure 7.

A second group of experiments considered deterministic transition timing. Because of the symmetric nature of the model its deterministic timing, the time gap between the local virtual time and the global virtual time within each LP appeared to be comparably small. In other words, the probability of a rollback is small in this cases, and rollback costs are much lower than with transitions with exponentially distributed enablings. The average time needed to perform a rollback on 32 processors decreases from  $\approx 7msec$  to  $\approx 2msec$  (The simulation of one event still needs about  $3msec$ ). The cost induced by the larger number of estimate function calls (each about  $140\mu sec$  on 32 processors), exceeded the gain in rollback performance, and a 7 % performance degrade over TW was observed. Finally, when transition timing was mixed, the performance gap between TW and the adaptive methods was not significant.

### 4.3 TW with explicit Throttling

The previous section demonstrated that TW could be accelerated by just starting a new loop (**continue** in step s9, Figure 1), instead of simulating every event immediately. This new start of the loop already represents an implicit throttle, with a delay depending on the number of received messages. (An additional bonus of this strategy is the possibility to detect stragglers earlier, since the input buffers are polled more frequently.)

Another variation of throttling is to explicitly delay the execution of events in step s9 of the adaptive TW engine (Figure 1). Figure 8 shows the acceleration gained when the CPU is delayed by  $3000\mu sec$  in step s9, before starting a new loop (Note that average event processing time varies between  $2000\mu sec$  and  $4000\mu sec$ , depending on the internal/external event ratio and the token load). Explicitly delaying yields an additional acceleration over TW (as in Figure 5, first row), but (slightly) degrades performance over the implicit throttling approach. An intuitive suggestion for the issue of finding an “appropriate” delay value is to adapt the explicit throttle to the load of an LP, since with a fixed explicit throttle, the blocking time is always either too short or too long. Experiments with a dynamic, periodic readjustments of the delay times (after  $n = 1, 10, 100$  transition firings) are also reported in Figure 8 (second row). Again, the acceleration for a high token load is little better than for plain TW, but still worse than the acceleration in the case of implicit throttling.

In conclusion, for the explicit throttling approach we can thus say that for the particular type of simulation models, and the target execution platform, it is more important to poll the input buffers for stragglers more frequently, i.e. to detect potential rollbacks earlier, than to idle for possible future stragglers.

## 5 Conclusions

We have extended the standard Time Warp parallel simulation technique by introducing an adaptive optimism control mechanism for the parallel simulation of spatially decomposed timed Petri nets. Our simulation engine, by temporarily blocking the processing of local transition firings, avoids the generation and send-out of token messages in states for which it is likely that they will have to be “rolled back”. A statistical analysis of the token message arrival history is used to make forecasts for the timestamps of future tokens, thus enabling every logical process to adapt its local synchronization behavior to the most efficient strategy with respect to the anticipated future.

Two classes of forecast methods were studied:

- For estimates based on (weighted) means, efficient (incremental) procedures can be implemented for next message timestamp forecast, causing negligible or minor intrusion on the simulation engine. Those methods (arithmetic mean, median and exponential smoothing), while improving TW performance significantly in the absence of trends and seasons in the arrival process, cannot cope well with nonstationary arrivals.
- At the cost of higher computational complexity, more sophisticated forecast methods with a much higher prediction precision in the case of periodic or seasonal (correlated) channel time increments can be used. We have proposed a time series analysis of the token time increments, and have developed forecast methods based on autoregressive moving average models and on Kalmanfilters. This approach can even further accelerate TW for “loaded” systems, i.e. where the message load is comparably high. It can, however, also tend to slow down TW in case there are not enough messages circulating among LPs, such that either no statistical significance can be achieved for the established forecast model, or the computational complexity of the model building and forecasting overwhelms the overall execution time gain.

For the parallel TW simulation of very large Petri net models we achieve convincing performance improvements, e.g. a 250 to 300 fold acceleration of the overall execution speed using 32 processors on the Meiko CS-2 multiprocessor as compared to executing the same model (using TW) on a single node. This effect mainly results from the decomposition of the Petri net model into smaller submodels which reduces memory access overhead at quadratic order. For Petri net models with high token populations, the adaptive protocols can further accelerate execution speed. Empirical evidence has been derived for the sensitivity of adaptive protocols to both the token load and the model timing.

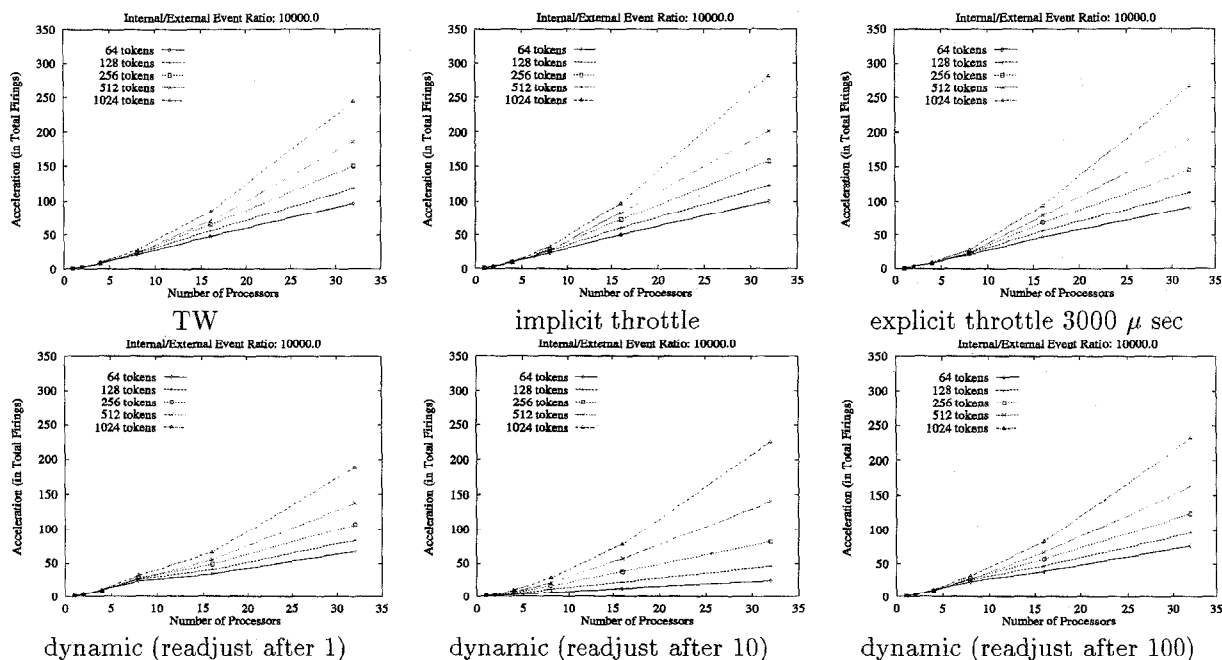


Figure 8: Document Model with an Internal/External Event Ratio of 10000

## References

- [1] F. Baccelli, N. Furmento, and B. Gaujal. Parallel and Distributed Simulation of Free Choice Petri Nets. In *Proceedings of the 9th Workshop on Parallel and Distributed Simulation (PADS '95)*, pages 3 – 10, 1995.
- [2] D. Ball and S. Hoyt. The Adaptive Time-Warp Concurrency Control Algorithm. In D. Nicol, editor, *Distributed Simulation. Proceedings of the SCS Multiconference on Distributed Simulation*, pages 174 – 177, San Diego, California, 1990. Society for Computer Simulation. Simulation Series, Vol. 22, No. 1.
- [3] G. E. P. Box and G. M. Jenkins. *TIME SERIES ANALYSIS forecasting and control*. Holden Day Series in Time Series Analysis. Holden Day, 2 edition, 1971.
- [4] P.J. Brockwell and R.A. Davis. *Time Series: Theory and Methods*. Springer Series in Statistics. Springer-Verlag, 2 edition, 1991.
- [5] G. Chiola and A. Ferscha. Distributed Simulation of Petri Nets. *IEEE Parallel and Distributed Technology*, 1(3):33 – 50, August 1993.
- [6] M. Q. Cui and St. J. Turner. A New Approach to the Distributed Simulation of Timed Petri Nets. In A. Javor, A. Lehmann, and I. Molnar, editors, *10<sup>th</sup> European Simulation Multiconference*, pages 90 – 94. SCS, 1996.
- [7] S. R. Das and R. M. Fujimoto. An Adaptive Memory Management Protocol for Time Warp Parallel Simulation. In *Proc. of the 1994 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, Nashville, 1994*, pages 201–210. ACM, 1994.
- [8] A. P. Dempster, N. M. Larid, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. R. Statist. Soc.*, (B 39):1–38, 1977.
- [9] A. Ferscha. Concurrent Execution of Timed Petri Nets. In J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila, editors, *Proceedings of the 1994 Winter Simulation Conference*, pages 229 – 236, 1994.
- [10] A. Ferscha. Probabilistic Adaptive Direct Optimism Control in Time Warp. In *Proceedings of the 9th Workshop on Parallel and Distributed Simulation (PADS '95)*, pages 120 – 129, 1995.
- [11] A. Ferscha. Parallel and Distributed Simulation of Discrete Event Systems. In A. Y. Zomaya, editor, *Parallel and Distributed Computing Handbook*, pages 1003 – 1041. McGraw-Hill, 1996.
- [12] A. Ferscha and G. Chiola. Self Adaptive Logical Processes: The Probabilistic Distributed Simulation Protocol. In *Proc. of the 27<sup>th</sup> Annual Simulation Symposium*, pages 78–88, Los Alamitos, California, 1994. IEEE Computer Society Press.
- [13] A. Ferscha and J. Lüthi. Estimating Rollback Overhead for Optimism Control in Time Warp. In *Proc. of the 28<sup>th</sup> Annual Simulation Symposium*, pages 2–12, Los Alamitos, California, 1995. IEEE Computer Society Press.
- [14] A. Ferscha and M. Richter. Massively Parallel Simulation of Business Process Models. In A. Javor, A. Lehmann, and I. Molnar, editors, *10<sup>th</sup> European Simulation Multiconference*, pages 377 – 381. SCS, 1996.
- [15] Donald O. Hamnes and Anand Tripathi. Investigations in Adaptive Distributed Simulation. In D. K. Arvind, Rajive Bagrodia, and Jason Yi-Bing Lin, editors, *Proceedings of the 8th Workshop on Parallel and Distributed Simulation (PADS '94)*, pages 20–23, July 1994.

- [16] A. H. Jazwinski. *Stochastic Process and Filtering Theory*. Academic Press, New York, 1970.
- [17] D. M. Nicol and S. Roy. Parallel Simulation of Timed Petri-Nets. In B. Nelson, D. Kelton, and G. Clark, editors, *Proceedings of the 1991 Winter Simulation Conference*, pages 574 – 583, 1991.
- [18] R. H. Shumway and D. S. Stoffer. An Approach to Time Series Smoothing and Forecasting using the EM Algorithm. *Journal of time Series Analysis*, 3(4):253–264, 1982.
- [19] G. S. Thomas and J. Zahorjan. Parallel Simulation of Performance Petri Nets: Extending the Domain of Parallel Simulation. In *Proc. of the 1991 Winter Simulation Conference*, 1991.

## Appendix A: Time Series Forecasting

This Appendix gives the statistical and algorithmic details for timestamp forecasting based on the consideration of  $(\delta_1, \delta_2, \dots, \delta_n)$  as realizations of random variables  $X_t, t \in T$ .

Generally when analyzing time series, a first insight into the dependencies among random variables is gained from covariance analysis. We briefly recall: If  $\{X_t, t \in T\}$  is a process such that  $Var(X_t) < \infty$  for each  $t \in T$ , then the **autocovariance function**  $\gamma_x(\cdot, \cdot)$  of  $\{X_t\}$  is defined by

$$\gamma_x(r, s) = Cov(X_r, X_s), \quad r, s \in T. \quad (7)$$

The process  $\{X_t, t \in \mathbb{Z}\}$ , with index set  $\mathbb{Z} = \{0, \pm 1, \pm 2, \dots\}$ , is said to be **stationary** if

- (i)  $E|X_t|^2 < \infty \quad \forall t \in \mathbb{Z}$ ,
  - (ii)  $EX_t = m \quad \forall t \in \mathbb{Z}$ ,
- and
- (iii)  $\gamma_x(r, s) = \gamma_x(r + t, s + t) \quad \forall r, s, t \in \mathbb{Z}$ .

If  $\{X_t, t \in \mathbb{Z}\}$  is stationary, then  $\gamma_x(r, s) = \gamma_x(r - s, 0)$  for all  $r, s \in \mathbb{Z}$ . It is therefore convenient to redefine the autocovariance function of a stationary process as the function of just one variable.

$$\gamma_x(h) \equiv \gamma_x(h, 0) = Cov(X_{t+h}, X_t) \quad t, h \in \mathbb{Z}. \quad (8)$$

The function  $\gamma_x(\cdot)$  will be referred to as the *autocovariance function* of  $\{X_t\}$  and  $\gamma_x(h)$  as its value at "lag"  $h$ . The **autocorrelation function (acf)** of  $\{X_t\}$  is defined analogously as the function whose value at lag  $h$  is [4]

$$\rho(h) \equiv \gamma_x(h) / \gamma_x(0) = Corr(X_{t+h}, X_t) \quad t, h \in \mathbb{Z}. \quad (9)$$

If  $\gamma(\cdot)$  is the autocovariance function of a stationary process  $\{X_t, t \in \mathbb{Z}\}$ , then

$$\gamma(0) \geq 0, \quad (10)$$

$$|\gamma(h)| \leq \gamma(0) \quad \forall h \in \mathbb{Z} \text{ and} \quad (11)$$

$$\gamma(h) = \gamma(-h) \quad \forall h \in \mathbb{Z}. \quad (12)$$

From the observation  $\{x_1, x_2, \dots, x_n\}$  of a stationary time series  $\{X_t\}$  we have to estimate the autocovariance function  $\gamma(\cdot)$  of the underlying process  $\{X_t\}$  to be able to construct an appropriate model for the message arrival pattern. The estimate of  $\gamma(\cdot)$  which we shall use is the *sample autocovariance function*.

**Definition A.1** The *sample autocovariance function* of  $\{x_1, x_2, \dots, x_n\}$  is defined [4]

$$\hat{\gamma}(h) = \frac{1}{n} \sum_{j=1}^{n-h} (x_{j+h} - \bar{x})(x_j - \bar{x}), \quad 0 \leq h \leq n, \quad (13)$$

and  $\hat{\gamma}(h) = \hat{\gamma}(-h)$ ,  $-n \leq h \leq 0$ , where  $\bar{x}$  is the sample mean  $\bar{x} = n^{-1} \sum_{j=1}^n x_j$ .

The divisor  $n$  is used rather than  $(n - h)$  since this ensures that the matrix  $\Gamma_n = [\hat{\gamma}(i - j)]_{i,j=1}^n$  is non-negative definite. In the future we will only consider centered time series  $\{X_t\}$  ( $\bar{x} = 0$ ) (i.e. transformed with respect to the series mean  $x_i = \delta_i - \mu$ ). The **partial autocorrelation (pacf)**, like the autocorrelation function, expresses essential information about the dependence structure of the stationary process. The partial autocorrelation  $\alpha(k)$  at lag  $k$  may be regarded as the correlation between  $X_1$  and  $X_{k+1}$ , after removing the effect of the intervening observation  $X_2, \dots, X_k$ . The pacf is obtained from [4]:

$$\begin{bmatrix} \rho(0) & \rho(1) & \dots & \rho(k-1) \\ \rho(1) & \rho(2) & \dots & \rho(k-2) \\ \vdots & \vdots & \ddots & \vdots \\ \rho(k-1) & \rho(k-2) & \dots & \rho(0) \end{bmatrix} \begin{bmatrix} \phi_{k1} \\ \phi_{k2} \\ \vdots \\ \phi_{kk} \end{bmatrix} = \begin{bmatrix} \rho(1) \\ \rho(2) \\ \vdots \\ \rho(k) \end{bmatrix}, \quad k \geq 1 \quad (14)$$

The partial autocorrelation  $\alpha(k)$  of  $\{X_t\}$  at lag  $k$  is  $\alpha(k) = \phi_{kk}$  for  $k \geq 1$ , where  $\phi_{kk}$  is uniquely determined by equation 14.

## Stationary ARMA Processes

Since the 1970 work by Box and Jenkins [3] autoregressive moving average (ARMA) models have become popular and important tool in the modeling and forecasting of time series data.

**Definition A.2 (The ARMA( $p, q$ ) Process)** The process  $\{X_t, t = 0, \pm 1, \pm 2, \dots\}$  is said to be an ARMA( $p, q$ ) process if  $\{X_t\}$  is stationary and for every  $t$ ,

$$X_t - \phi_1 X_{t-1} - \dots - \phi_p X_{t-p} = Z_t - \theta_1 Z_{t-1} - \dots - \theta_q Z_{t-q} \quad (15)$$

where  $\{Z_t\} \sim WN(0, \sigma^2)$ .

Using the backward shift operator, the equation (15) can be written in the more compact form

$$\phi(B)X_t = \theta(B)Z_t, \quad t = 0, \pm 1, \pm 2, \dots, \quad (16)$$

where  $\phi$  and  $\theta$  are polynomials at degree  $p$  and  $q$  respectively:

$$\phi(z) = 1 - \phi_1 z - \dots - \phi_p z^p \quad (17)$$

$$\theta(z) = 1 + \theta_1 z + \dots + \theta_q z^q, \quad (18)$$

and the backward shift operator  $B$  is defined by

$$B^j X_t = X_{t-j}, \quad j = 0, \pm 1, \pm 2, \dots \quad (19)$$

A process is called a *moving average process* of order  $q$  (denoted as MA( $q$ )), if  $\phi(z) \equiv 1$ :

$$X_t = \theta(B)Z_t \quad (20)$$

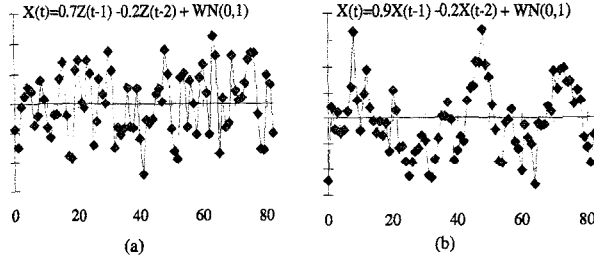


Figure 9: (a) An Moving Average (MA(2)) Process and (b) an Autoregressive (AR(2)) Process

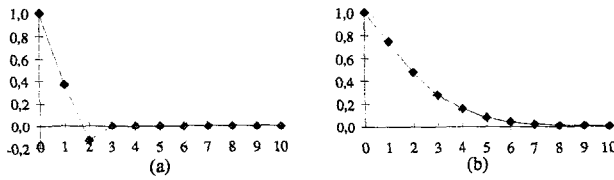


Figure 10: Autocorrelation of (a) an MA(2)-Process and (b) an AR(2) Process

A realization of  $\{X_1, \dots, X_{90}\}$  of an MA(2)  $X_t = 0.7Z_{t-1} - 0.2Z_{t-2} + WN(0, 1)$  is show in Figure 9a.

A process is called an *autoregressive process* of order  $p$  (denoted as AR( $p$ )), if  $\theta(z) \equiv 1$ :

$$\phi(B)X_t = Z_t. \quad (21)$$

As an example, a realization  $\{X_1, \dots, X_{90}\}$  of an AR(2)  $X_t = 0.9X_{t-1} - 0.2X_{t-2} + WN(0, 1)$  is show in Figure 9b.

It can be shown that for every model (15) a representation exists as either an **infinite AR** (AR( $\infty$ )), or an **infinite MA** (MA( $\infty$ )) process. The model can be written either as

$$\sum_{j=0}^{\infty} \pi_j X_{t-j} = \pi(B)X_t = Z_t \quad (22)$$

or as

$$X_t = \sum_{j=0}^{\infty} \psi_j Z_{t-j} = \psi(B)Z_t \quad (23)$$

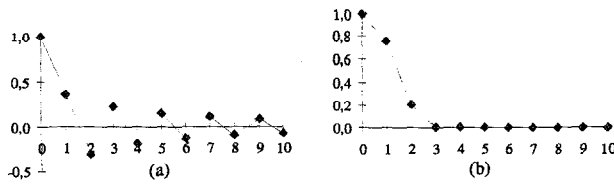


Figure 11: Partial Autocorrelation of (a) an MA(2)-Process and (b) an AR(2) Process

Where

$$\pi(z) = \sum_{j=0}^{\infty} \pi_j z^j = \phi(z)/\theta(z) \quad |z| \leq 1 \quad (24)$$

$$\psi(z) = \sum_{j=0}^{\infty} \psi_j z^j = \theta(z)/\phi(z) \quad |z| \leq 1 \quad (25)$$

are an infinite series of constants  $\{\pi_j\}$  ( $\{\psi_j\}$ ) such that  $\sum_{j=0}^{\infty} |\pi_j|$  ( $\sum_{j=0}^{\infty} |\psi_j|$ )  $< \infty$ .

## Prediction of Stationary Processes

The idea to predict the values  $\{X_t, t \geq n+1\}$  of a stationary time stamp increment process based on  $\{X_1, \dots, X_n\}$  is to use the observations of this process made so far. Besides the direct computation of the predictors, where a large system of linear equations has to be solved [4], a recursive method exists for the one-step predictors  $\hat{X}_{n+1}, n \geq 1$ . Given the observations from a zero-mean stationary time series, with  $\hat{\gamma}(0) > 0$ , we can fit an autoregressive process of order  $m < n$  to the data using the *Durbin-Levinson* algorithm. The fitted AR( $m$ ) process in this case is

$$X_t - \hat{\phi}_{m1}X_{t-1} - \dots - \hat{\phi}_{mm}X_{t-m} = Z_t \quad \{Z_t\} \sim WN(0, \hat{v}_m) \quad (26)$$

which can be determined for  $m = 1, 2, \dots, n-1$  recursively from the relations,  $\hat{\phi}_{11} = \hat{\rho}(1)$ ,  $\hat{v}_0 = \hat{\gamma}(0)$ ,

$$\hat{\phi}_{mm} = \left[ \hat{\gamma}(m) - \sum_{j=1}^{m-1} \hat{\phi}_{m-1,j} \hat{\gamma}(m-j) \right] / \hat{v}_{m-1}, \quad (27)$$

$$\begin{bmatrix} \hat{\phi}_{m1} \\ \hat{\phi}_{m2} \\ \vdots \\ \hat{\phi}_{m,m-1} \end{bmatrix} = \begin{bmatrix} \hat{\phi}_{m-1,1} \\ \hat{\phi}_{m-1,2} \\ \vdots \\ \hat{\phi}_{m-1,m-1} \end{bmatrix} - \hat{\phi}_{m,m} \begin{bmatrix} \hat{\phi}_{m-1,m-1} \\ \hat{\phi}_{m-1,m-2} \\ \vdots \\ \hat{\phi}_{m-1,1} \end{bmatrix} \quad (28)$$

$$\hat{v}_m = \hat{v}_{m-1} (1 - \hat{\phi}_{mm}^2), \quad (29)$$

where the  $\hat{\phi}_{11}, \hat{\phi}_{22}, \dots$  are estimates for the partial autocorrelation  $\hat{\alpha}$  at lags 1, 2,  $\dots$

Alike the Durbin Levinson Algorithm for fitting autoregressive models, there is a recursive way of fitting moving average models

$$X_t = Z_t - \hat{\theta}_{m1}Z_{t-1} - \dots - \hat{\theta}_{mm}Z_{t-m} \quad \{Z_t\} \sim WN(0, \hat{v}_m) \quad (30)$$

of orders  $m = 1, 2, \dots$  by means of the *innovations algorithm*. The estimates of  $\hat{\theta}_m$  and white noise variance  $\hat{v}_m, m = 1, 2, \dots$  are obtained as follows: If  $\hat{\gamma}(0) > 0$ , we define the estimates  $\hat{\theta}$  and  $\hat{v}_m$  in equation (30) for  $m = 1, 2, \dots$ , by the recursion,  $\hat{v}_0 = \hat{\gamma}(0)$ ,

$$\hat{\theta}_{m,m-k} = \frac{1}{\hat{v}_k} \left[ \hat{\gamma}(m-k) - \sum_{j=0}^{k-1} \hat{\theta}_{m,m-j} \hat{\theta}_{k,k-j} \hat{v}_j \right], \quad k = 0, \dots, m-1 \quad (31)$$

and

$$\hat{v}_m = \hat{\gamma}(0) - \sum_{j=0}^{m-1} \hat{\theta}_{m,m-j}^2 \hat{v}_j. \quad (32)$$

**Preliminary Estimation for ARMA( $p, q$ ) Processes** If  $\{X_t\}$  is an zero-mean ARMA( $p, q$ ) process,

$$X_t - \phi_1 X_{t-1} - \dots - \phi_p X_{t-p} = Z_t - \theta_1 Z_{t-1} - \dots - \theta_q Z_{t-q}$$

$$\{Z_t\} \sim WN(0, \sigma^2) \quad (33)$$

then (5) can be written as an infinite MA model

$$X_t = \sum_{j=0}^{\infty} \psi_j Z_{t-j}.$$

We can also rewrite (25), such that the coefficients  $\psi_j$  satisfy

$$\begin{cases} \psi_0 = 1, \\ \psi_j = \theta_j + \sum_{i=1}^{\min(j,p)} \phi_i \psi_{j-i}, \quad j = 1, 2, \dots \end{cases} \quad (34)$$

and by convention  $\theta_j = 0$  for  $j > q$  and  $\phi_j = 0$  for  $j > p$ . To estimate  $\psi_1, \dots, \psi_{p+q}$ , we can use the innovation algorithm estimates  $\hat{\theta}_{m1}, \dots, \hat{\theta}_{m,p+q}$  of an MA( $m$ ) ( $p+q \ll m \approx n^{1/3}$ ). Replacing  $\psi_j$  by  $\hat{\theta}_{mj}$  in (34) and solving the resulting equations,

$$\hat{\theta}_{mj} = \theta_j + \sum_{i=1}^{\min(j,p)} \phi_i \hat{\theta}_{m,j-i}, \quad j = 1, 2, \dots, p+q, \quad (35)$$

we obtain initial parameter estimates for  $\phi_i$  ( $1 \leq i \leq p$ ) and  $\theta_j$  ( $1 \leq j \leq q$ ). From equation (35) with  $j = q+1, \dots, q+p$ , we see that  $\hat{\phi}_i$  should satisfy the equation:

$$\begin{bmatrix} \hat{\theta}_{m,q+1} \\ \hat{\theta}_{m,q+2} \\ \vdots \\ \hat{\theta}_{m,q+p} \end{bmatrix} = \begin{bmatrix} \hat{\theta}_{m,q} & \hat{\theta}_{m,q-1} & \dots & \hat{\theta}_{m,q+1-p} \\ \hat{\theta}_{m,q+1} & \hat{\theta}_{m,q} & \dots & \hat{\theta}_{m,q+2-p} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\theta}_{m,q+p-1} & \hat{\theta}_{m,q+p-2} & \dots & \hat{\theta}_{m,q} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_p \end{bmatrix} \quad (36)$$

After solving equation (36) for  $\phi_i$  ( $1 \leq i \leq p$ ) the estimates of  $\theta_j$  ( $1 \leq j \leq q$ ) are obtained from

$$\hat{\theta}_j = \theta_{mj} - \sum_{i=1}^{\min(j,p)} \hat{\phi}_i \hat{\theta}_{m,j-i}, \quad j = 1, 2, \dots, q. \quad (37)$$

The white noise variance  $\sigma^2$  is estimate by  $\sigma^2 = \hat{v}_m$ .

**Maximum Likelihood Estimation for ARMA Processes** The preliminary estimates of  $\theta_j$  and  $\phi_j$  are now used as an initial start value for the non-linear optimization carried out in the course of maximizing the maximum likelihood. The one-step predictor  $\hat{X}_{n+1}$  of an ARMA( $p, q$ ) is given by

$$\hat{X}_{n+1} = \begin{cases} \sum_{j=1}^n \theta_{nj} (X_{n+1-j} - \hat{X}_{n+1-j}), & 1 \leq i < m = \max(p, q), \\ \sum_{j=1}^p \phi_j X_{n+1-j} + \sum_{j=1}^q \theta_{nj} (X_{n+1-j} - \hat{X}_{n+1-j}), & i \geq m. \end{cases} \quad (38)$$

and the likelihood of the observation  $(X_1, X_2, \dots, X_n)$  is

$$L(\vec{\theta}, \vec{\phi}, \sigma^2) = (2\pi)^{-n/2} (v_0 v_1 \dots v_{n-1})^{-1/2} \exp \left[ -\frac{1}{2} \sum_{j=1}^n (X_j - \hat{X}_j)^2 / v_{j-1} \right]. \quad (39)$$

Criteria have been developed (e.g. Akaike's BIC and AIC criterion) attempting to prevent from overfitting by effectively assigning a cost to the introduction of each additional parameter. We use the AICC criterion defined by:

$$AICC(\vec{\theta}, \vec{\phi}) = -2 \ln L(\vec{\theta}, \vec{\phi}, \sigma^2) + \frac{2n(p+q+1)}{(n-p-q-2)} \quad (40)$$

## Identification Techniques

The problem of identifying the appropriate ARMA( $p, q$ ) model order to represent the time series  $\{X_t\}$  is to determine  $p$  and  $q$ . For an pure moving average process order identification is simple: the autocovariance function of the MA( $q$ ) process

$$X_t = \sum_{j=0}^q \theta_j Z_{t-j},$$

has the form (as depicted in Figure 10 (a)):

$$\gamma(k) = \begin{cases} \sigma^2 \sum_{j=0}^q \theta_j \theta_{j+|k|}, & |k| \leq q, \\ 0, & |k| > q, \end{cases} \quad (41)$$

where  $\theta_0$  is defined to be 1 and  $\theta_j$ ,  $j > q$ , is defined to be zero.

For an pure autoregressive process AR( $p$ ) the order  $p$  can be obtained from the cut off in the partial autocorrelation function  $\alpha(\cdot)$  (see Figure 11 (b)):  $p$  could be easily determined as we know that

$$\alpha(k) = \begin{cases} \phi_{kk} & |k| \leq p, \\ 0, & |k| > p. \end{cases} \quad (42)$$

In contrast to the partial autocorrelation function of an AR( $p$ ) process that of an MA( $q$ ) process does not vanish for large lags. It is however bounded in absolute value by a geometrically decreasing function [4] (see Figure 11 (a)).

For fitting an AR( $p$ ) or MA( $q$ ) to an observed sample, the sample (partial-) autocorrelation  $\alpha(m)$  or  $\gamma(m)$  is not zero for  $m > p$  or  $q$ . If the order of the process is  $p$  or  $q$ , then for  $m > p$ ,  $\alpha(m)$  or  $m > q$ ,  $\gamma(m)$  will fall between the bounds  $\pm 1.96n^{1/2}$ . If the underlying process is ARMA( $p, q$ ), then the model order identification requires the investigation of all meaningful pairs ( $p, q$ ) and choose that pair which minimizes one of the criteria given above. Pragmatically we first compute the AICC criterion with the preliminary estimates obtained by (36) and (37) for ARMA( $p_0, p_0$ ) models with  $p_0 = 1, 2, \dots$ . Then we try to eliminate one or more coefficients of the ARMA( $p_0 = p, p_0 = q$ )-model in order to minimize AICC.

## Appendix B: Kalmanfilter Forecasting

This method assumes that the series of interest  $\{X_t\}$  is not observed directly, but only as component in the random regression model

$$y_t = M\bar{x}_t + v \quad t = 1, 2, \dots, n \quad (43)$$

where  $M$  is a know  $1 \times p$  design vector which expresses the pattern that converts the unobserved stochastic vector  $\bar{x}_t$  into the observed series  $y_t$ .  $v$  is a zero-mean normally distributed noise vector. The random series  $\bar{x}_t$  is modeled as a multivariate process of the form

$$\bar{x}_t = \Phi\bar{x}_{t-1} + \bar{w}_t \quad t = 1, 2, \dots, n \quad (44)$$

where  $\Phi$  is a  $p \times p$  transition matrix describing the way the underlying series moves through successive time periods.  $\{X_t\}$  does not need to be stationary. The initial value  $\bar{x}_0$  is assumed to be a normal random vector with the  $p \times p$  covariance matrix  $\Sigma$ .  $\bar{w}_t$  is a  $p \times 1$  noise vector with zero-mean uncorrelated normal distributed terms and with common covariance matrix  $Q$ .

The model defined by equation (43) and (44) is a generalization of the ordinary AR(p) model in (21). A recursive method using the EM (*Expectation Maximization*) algorithm described in (e.g. [8]) has been proposed by Shumway and Stoffer [18] to compute the Kalman filter estimators, which is explained in the sequel.

The log likelihood of the complete data  $\bar{x}_0, \bar{x}_1, \dots, \bar{x}_n, \bar{y}_1, \dots, \bar{y}_n$  can be written in the form

$$\begin{aligned} \log L = & -\frac{1}{2} \log |\Sigma| - \frac{1}{2} (\bar{x}_0)' \Sigma^{-1} (\bar{x}_0) \\ & -\frac{n}{2} \log |Q| - \frac{1}{2} \sum_{t=1}^n (\bar{x}_t - \Phi\bar{x}_{t-1})' Q^{-1} (\bar{x}_t - \Phi\bar{x}_{t-1}) \\ & -\frac{n}{2} \log |v| - \frac{1}{2} \sum_{t=1}^n (\bar{y}_t - M\bar{x}_t)' v^{-1} (\bar{y}_t - M\bar{x}_t) \end{aligned} \quad (45)$$

The log likelihood given in (46) depends on the unobserved data series  $\bar{x}_0, \bar{x}_1, \dots, \bar{x}_n$ . We apply the EM algorithm on the observed series  $\bar{y}_0, \bar{y}_1, \dots, \bar{y}_n$  to maximize the log likelihood with respect to the parameters  $\Sigma, \Phi, Q, v$ . That is, determine the estimated parameters in iteration  $(r+1)$  as the values  $\Sigma, \Phi, Q$  and  $v$  which maximize

$$G(\Sigma, \Phi, Q, v) = E_r(\log L | \bar{y}_0, \bar{y}_1, \dots, \bar{y}_n) \quad (46)$$

where  $E_r$  denotes the conditional expectation relative to a density containing the  $r$ -th iteration values  $\Sigma_{(r)}, \Phi_{(r)}, Q_{(r)}$ , and  $v_{(r)}$  [18]. Rewriting equation (46) using

$$\bar{x}_t^n = E(\bar{x}_t | \bar{y}_1, \dots, \bar{y}_n), \quad (47)$$

and the covariance Functions

$$P_t^n = \text{cov}(\bar{x}_t | \bar{y}_1, \dots, \bar{y}_n) \quad (48)$$

and

$$P_{t,t-1}^n = \text{cov}(\bar{x}_t, \bar{x}_{t-1} | \bar{y}_1, \dots, \bar{y}_n) \quad (49)$$

yields

$$G(\Sigma, \Phi, Q, v) = -\frac{1}{2} \log |\Sigma| - \frac{1}{2} \text{trace} \left\{ \Sigma^{-1} (P_0^n + (\bar{x}_0^n)(\bar{x}_0^n)') \right\}$$

$$\begin{aligned} & -\frac{n}{2} \log |Q| - \frac{1}{2} \text{trace} \left\{ Q^{-1} (C - B\Phi' - \Phi B' + \Phi A\Phi') \right\} \\ & -\frac{n}{2} \log |v| \end{aligned} \quad (50)$$

$$-\frac{1}{2} \text{trace} \left\{ \frac{1}{v} \sum_{t=1}^n [(\bar{y}_t - M\bar{x}_t^n)(\bar{y}_t - M\bar{x}_t^n)' + MP_t^n M'] \right\}$$

where

$$A = \sum_{t=1}^n (P_{t-1}^n + \bar{x}_{t-1}^n \bar{x}_{t-1}^{n'}) \quad (51)$$

$$B = \sum_{t=1}^n (P_{t,t-1}^n + \bar{x}_t^n \bar{x}_{t-1}^{n'}) \quad (52)$$

and

$$C = \sum_{t=1}^n (P_t^n + \bar{x}_t^n \bar{x}_t^{n'}) \quad (53)$$

The Kalman filter terms  $\bar{x}_t^n, P_t^n$  and  $P_{t,t-1}^n$  are computed under the parameter values of  $\Phi_{(r)}, Q_{(r)}$  and  $v_{(r)}$  ( $\Sigma$  is fixed at a reasonable baseline level) using the recursions given by [16] pp. 201, 217 and [18] pp. 263.

For  $t = 1, \dots, n$

$$P_t^{t-1} = \Phi_{(r)} P_{t-1}^{t-1} \Phi_{(r)}' + Q_{(r)} \quad (54)$$

$$K_t = P_t^{t-1} M' (M P_t^{t-1} M' + v_{(r)})^{-1} \quad (55)$$

$$P_t^t = P_t^{t-1} - K_t M P_t^{t-1} \quad (56)$$

$$\bar{x}_t^{t-1} = \Phi_{(r)} \bar{x}_{t-1}^{t-1} \quad (57)$$

$$\bar{x}_t^t = \bar{x}_t^{t-1} + K_t (\bar{y}_t - M \bar{x}_t^{t-1}) \quad (58)$$

where we take  $\bar{x}_0^0 = \bar{0}$  and  $P_0^0 = \Sigma$ . In order to calculate  $\bar{x}_t^n$  and  $P_t^n$  one performs the set of backward recursions (for  $t = n, n-1, \dots, 1$ ) on the equations

$$J_{t-1} = P_{t-1}^{t-1} \Phi_{(r)}' (P_t^{t-1})^{-1} \quad (59)$$

$$\bar{x}_{t-1}^n = \bar{x}_t^{t-1} + J_{t-1} (\bar{x}_t^n - \Phi_{(r)} \bar{x}_{t-1}^{t-1}) \quad (60)$$

$$P_{t-1}^n = P_{t-1}^{t-1} + J_{t-1} (P_t^n - P_t^{t-1}) J_{t-1}' \quad (61)$$

and  $P_{t,t-1}^n$  can be calculated using the backward recursion (for  $t = n, n-1, \dots, 2$ ) where

$$P_{n,n-1}^n = (I - K_n M) \Phi_{(r)} P_{n-1}^{n-1} \quad (62)$$

$$P_{t-1,t-2}^n = P_{t-1}^{t-1} J_{t-2}' + J_{t-1} (P_{t,t-1}^n - \Phi_{(r)} P_{t-1}^{t-1}) J_{t-2}' \quad (63)$$

We get the next iteration of  $\Phi_{(r+1)}, Q_{(r+1)}$  and  $v_{(r+1)}$  if we set

$$\Phi_{(r+1)} = B A^{-1}, \quad (64)$$

$$Q_{(r+1)} = n^{-1} (C - B A^{-1} B') \quad (65)$$

and

$$v_{(r+1)} = \frac{1}{n} \sum_{t=1}^n [(\bar{y}_t - M \bar{x}_t^n)(\bar{y}_t - M \bar{x}_t^n)' + M P_t^n M'] \quad (66)$$

which maximize the last two lines in the likelihood function (51).

Now the vector  $\bar{x}_t^n$  for  $t > n$  is the forecast value for the underlying series.