2. Script-based knowledge representation

A script is a knowledge representation scheme for representing time-based events. "A script is a finite set of events of some duration and importance oriented towards a goal." [17]. Schank and Abelson [18] state "... a script, ... is a stereotypical representation of a sequence of actions oriented toward attaining some goal." A script-based knowledge representation in an expert system stores data about sequences of events with the following characteristics; *duration:* a start and a finish; *dependency:* certain events occur before/after others, and *stereotypical aspects:* event sequences are often predictable and well-understood.

The script-based knowledge representation is a specific application of the frame knowledge representation. Each frame's slot definitions are constructed to apply to the specific application under consideration [21].

Scripts were initially formulated, [18], [19] and [20], to provide structures and manipulative ability for the handling of time and dependency-based events in the domain of "understanding" natural language. This work established conceptual dependency operators within natural language as a formal methodology for representing the semantic content of sentences and stories. Scripts are suitable for describing, storing, and searching for sequences of actions with attendant states. Script structures and manipulations have been applied to a variety of contexts. See, for example, [4], [9], [11], [14], [15] and [16].

3. Representing locking event sequences as scripts

Locking event sequences exhibit, *inter alia*, the same characteristics as scripts. Each locking event (activity) has starting and finishing times; each event (except the first and last events) has a predecessor(s) and a successor(s); and, the combination of events in sequence usually forms a well-ordered, stereotypical and understood series of events. In the EAGLE context, we regard the "goal" of these locking events sequences as the attainment of deadlock. EAGLE formats locking event sequences as scripts, and stores those scripts which resulted in deadlock.

Each frame (script) in DDA contains slots for:

> script name, process roles, resource props, expected event, critical event, critical event response, sequence description, response, activation level, utilisation level, minimum activation level and maximum activation level.

The *script name* uniquely identifies each script. The *process roles* and *resource props* identify the number of transactions and resources involved in the deadlock. The *expected event* is used to look for the next expected event for a script. Any OBJECTion to the granting of a lock request must take place before the *critical event*. The *sequence description* describes a stereotypical event sequence and the *response* is the result of a match against this sequence description. The various *activation* and *utilisation levels* monitor and control the activation of the script.

4. EAGLE Implementation

The EAGLE system was developed in the Clipper and Common LISP. Clipper was used for the simulation of database activities, and Common LISP was used for the Matcher/Analyser which generates stereotypical event sequences and compares locking event sequences against stored stereotypical event sequences. EAGLE is the script-based implementation of the DDA scheme. Incoming lock requests (as part of locking event sequences) are passed from the lock manager (LOMAN) to EAGLE, where they are matched against stored stereotypical locking event sequences, representing various patterns of lock events which lead to deadlock situations. Should an event (lock or unlock request) in a locking event sequence cause a match to occur with a stored stereotype script, the lock or unlock request is rejected. If no match occurs, the lock event is approved. Fig. 1 below shows an overview of the EAGLE operating environment:

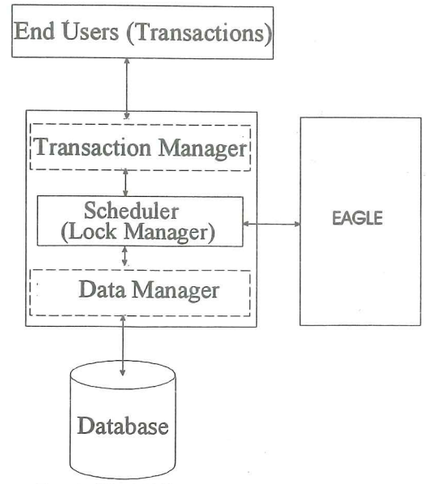


Fig. 1. The EAGLE operating environment

LOMAN receives transactions from the Transaction Manager. LOMAN checks its lock table (if a lock request is received) to determine if the requested resource is available, *i.e.* the resource is not currently locked by another transaction. If the lock request is approved by LOMAN, the lock request details (transaction and resource(s) involved) are passed to EAGLE for further inspection. EAGLE adds the current lock request to the script for that transaction. EAGLE then matches the current content of the locking event sequence against stereotypical scripts to estimate the probability of a future deadlock occurring.

EAGLE calculates a similarity metric (SM) or belief [17] for each instantiated script to determine the degree of relevance of a script to an observed locking event sequence. This is similar to Benoit et al's [4] SCAN system, where an overall likelihood is computed for each of the scripts. EAGLE either denies or approves the lock request based on the computed SM. If denied, the transaction is forced to re-request the lock, at which time, if LOMAN approves the request, EAGLE will re-evaluate the lock request.

The EAGLE system initially has no stored scripts describing deadlock situations stored. As a result, all "early" lock requests are granted. When a deadlock occurs, the contents of that locking event sequence are passed to the Learner sub-system of EAGLE. Here, the current locking event sequence is converted into both a *specific* stereotypical event sequence (one which contains lock, unlock, and wait tasks) and into a *generic* event sequence (one which contains lock and wait tasks of only the transactions involved in the deadlock). As further deadlocks occur, so the number of stored scripts increases, building a Script Base. The Matcher inside EAGLE continues to match fresh locking event sequences against the Script Base. If a match is made, EAGLE OBJECTs to the current request, and returns that OBJECTion to LOMAN. EAGLE has **four main components**, as depicted below in Fig 2..

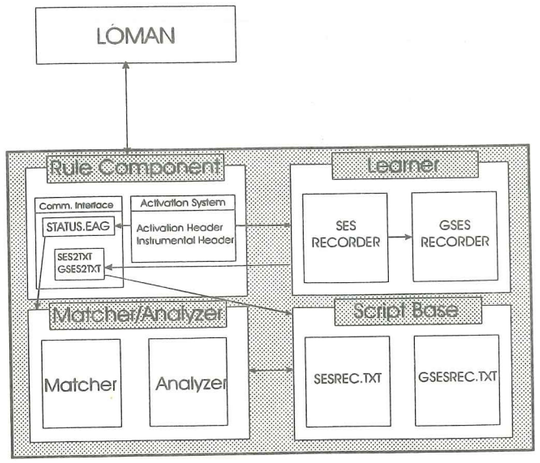


Fig. 2 EAGLE's components

Consider each component in turn;

* **Rule Component.** The rule component has two sub-components, namely the Communication Interface and the Activation System. The Communication Interface is responsible for passing messages and responses between LOMAN, the Matcher/Analyser and the Script Base. The Activation System is responsible for activating the Matcher/Analyser when the Precondition Header (at least 2 concurrent processes in operation) and the Instrumental Header (at least one lock request for a resource required by another transaction) are satisfied.

* **Matcher/Analyser Component.** The second component of EAGLE is the Matcher/Analyser, based on the IPS architecture of [4]. The Matcher/Analyser is activated by the Rule Component. The Matcher/Analyser receives the current locking event sequence from the Communication Interface of the Rule Component. The Matcher/Analyser compares the current locking event sequence against stereotypical locking event sequences stored in the Script Base. If the conditions for an OBJECTion are met, the Matcher/Analyser will return an OBJECT response to the Rule Component, and the transaction lock request is not granted. Fig. 3 below depicts an OBJECT being made based on a successful script instantiation. If no other transaction has made any requests, the same transaction may attempt to re-request the resource. This component also converts actual locking event sequences into stereotypical format.

```
(T02 + R01)(T01 - R01)(T03 * R01)(T03 + R02)
(T02 - R02)(T03 * R02)(T03 - R01)(T03 - R02)
(T04 * R02)(T04 * R01)(T02 + R02)(T04 - R02)
)

OBJECT!!!
Match against Specific scripts occured.
Sequence of events:
((T01 * R02)(T01 * R01)(T03 + R01)(T01 - R02)(T02 * R02)
(T02 + R01)(T01 - R01)(T03 * R01)(T03 + R02)
(T02 - R02)(T03 * R02)(T03 - R01)(T03 - R02)
(T04 * R02)(T04 * R01)(T02 + R02)(T04 - R02)
)
Name of script: S_P2R2_6
Activation level: 0.33
CE of the script: (LOCK ?PD ?RB)
Script sequence:
((LOCK ?PA ?RA)(LOCK ?PA ?RB)(WAIT ?PB ?RA)(UNLK ?PA ?RA)(LOCK ?PB ?RA)
(WAIT ?PC ?RA)(WAIT ?PD ?RB)(WAIT ?PB ?RB)(UNLK ?PA ?RB)
(LOCK ?PD ?RB)(WAIT ?PD ?RA))
Bound stereotype:
((T04 * R02)(T04 * R01)(T02 + R02)(T04 - R02)(T02 * R02)
(NIL + R02)(NIL + R01)(T02 + R01)(T04 - R01)
(NIL * R01)(NIL + R02))
SM: 4/11
```

Fig. 3. EAGLE OBJECTing to a lock request

* **Script Base Component.** The third component of EAGLE is the Script Base. The Script Base holds the stereotypical locking event sequences which have been encountered previously. There are two types of stereotypes, namely, a *specific* stereotype and a *generic* stereotype. *A Specific Stereotype* describes a stereotypical event sequence in the context of the current operational environment, *i.e.* a specific stereotype is not applicable in another context. A specific stereotype records all lock, unlock, and wait events of deadlock participants and non-deadlock participants. *A Generic Stereotype* describes scripts in a domain-independent context. No details of non-participating transactions' events or unlock requests are recorded in the generic stereotype. Generic stereotypes are useful in situations where specific stereotypes have not yet been defined.

* **Learner Component.** The Learner component is activated by the Rule Component whenever LOMAN's deadlock detector detects a deadlock. To detect deadlocks, LOMAN constructs wait-for-graphs which are analysed for cycles showing the presence of deadlock. The deadlock detection technique is similar to Topological Sorting [3], using a graph reduction approach with an adjacency list representation of the graph. The entire list is scanned for cycles. The details of the event sequence leading to the detected deadlock are converted into specific and generic stereotypical forms and stored in the Script Base. The utilisation of the scripts recorded in the Script Base is continually monitored. The Activation Level of the script is adjusted to tune the script to an effective level of performance. Over-utilisation of a script can lead to unnecessary system interference, and under-utilisation of a script wastes system resources. Scripts failing to perform within the defined parameters, after adjustments have been made, are removed from the Script Base.

5. Test runs

A simulated transaction environment was developed in which EAGLE operated. A series of simulated transactions was run in order to test and evaluate EAGLE's impact on the occurrence of deadlock. Simulated transactions occurred in two situations, *viz*, (i) EAGLE active, *i.e.* Dynamic Deadlock Avoidance (DDA), and (ii) EAGLE inactive, *i.e.* Deadlock Detection (DLD). [5] identified the following assumptions for evaluating a concurrency control algorithm, *viz*, "All transactions require the same number of locks, all data items are accessed with equal probability, and all locks are write locks. The transactions use Strict 2PL: data items are locked before they are accessed, and locks are released only after all transactions commit (or abort). The database is centralised.....". See [8] for discussion of these assumptions.

5.1 Simulation parameters

Several parameters were used within the transaction simulations. Considering each parameter in turn:

* **Transaction Size.** This is the number of resources processed by transactions. Except for run 8, all transactions accessed the same number of resources. Varying the transaction size alters the number of possible deadlock situations, and hence the number of scripts required to represent the stereotypical deadlock situations.

* **Multiprogramming Level (MPL).** The MPL is the number of transactions that are processed concurrently in a run. The higher the MPL the higher the potential resource conflict, and consequently the higher the probability of deadlock. The MPL parameter varies from 2 to 5 transactions.

* **Resource Contention**. This is the proportion of the resources used by all participating transactions. A value of 1.0 means that all transactions require all resources, resulting in higher resource contention. A value of 0.5 means that only half of the resources required are common to all transactions, so resulting in lower resource contention. This parameter allows for larger transaction sizes with lower resource contention. This parameter was set to 1.0 except in Run 8 and Run 12. In Run 8 it was set at 0.5 and in Run 12 to 0.2. This allowed for a low conflict situation between 4 concurrent transactions.

* **Activation Level.** This is the degree of similarity required (measured between 0, no similarity, and 1, exact match) between an observed locking event sequence and a stored script to activate the script. The activation level of all new scripts is initially at 0.5. This level is adjusted as scripts are utilised.

* **Clean Level.** This parameter determines when un/under-utilised scripts are removed from the Script Base. The clean level is the number of activations of the Matcher/Analyser after which un/under-utilised scripts are removed. The clean level was set at 200. Decreasing the level below 200 caused many scripts to be removed from the database that had not yet been given sufficient opportunity to be activated. Increasing the clean level above 200 caused the number of scripts in the Script Base to grow too large, thus resulting in long matching times and multiple OBJECT ceilings (see below). The clean level was set at 200.

* **OBJECT Ceiling.** In order to avoid an EAGLE-lock (deadlock-like) situation arising, a limit was set for the number of consecutive OBJECTions. If no such limit is set, it is possible that no transactions will proceed, because all transactions match a script in the Script Base. The OBJECT ceiling limits the number of consecutive OBJECTions that will be permitted. Once this ceiling value is reached, any further lock request is automatically granted. Setting the OBJECT ceiling too low *e.g.* 2, means that many valid OBJECTions are ignored. However, setting the OBJECT ceiling too high *e.g.* 10, slows the forward progress of transactions. An OBJECT ceiling of 5 was found to be an effective value, providing a trade-off between ignoring valid OBJECTions and impeding transaction progress.

Fig. 4 below summarises the main **simulation parameters** used:

TRAN_SIZE	The resource size of a transaction, varies from 2 to 5.
MPL	Multi-programming level, varies from 2 to 5 transactions.
RESOURCE CONTENTION	Proportion of resources used by all participating transactions, varies from 0.2 to 1.0.
ACTLEVEL	Activation Level - The degree of similarity necessary to activate a script. The initial ActLevel is 0.50.
OBJCEIL	The OBJECT ceiling is the maximum number of consecutive OBJECTions allowed. Set at 5.

Fig. 4. EAGLE simulation run parameters.

5.2 Test data

Twelve sets of test data were used, with varying MPLs and TRAN_SIZEs. As performance was not being measured, it was not necessary for transactions to perform any actual processing/updating of a real database. Simulated transactions acquire locks and then, on completion, release all acquired locks. The MPL varies between 2 and 5 concurrent transactions, and the TRAN_SIZE between 2 and 5 resources. These values were chosen for the following reasons:

* **Simulation Time.** The runs were constructed to test the algorithm under varying degrees of resource contention. The MPL and TRAN_SIZEs were therefore "selected so as to jointly yield a region of operation which allows the interesting performance effects to be observed without necessitating impossibly long simulation times" [1]. This is particularly true in the simulated environment, when DDA is active, as all lock requests have to be passed to the Matcher/Analyser for evaluation. As the Matcher/Analyser was implemented in interpreted LISP, increasing the MPL or TRAN_SIZE results in more complex matches and extremely long elapsed times for simulation runs.
* **Rarity of Conflict.** Besides the problem of long simulation run times, with high MPLs and large TRAN_SIZEs, "conflicts become very rare; and when conflicts are rare, all concurrency control algorithms perform alike" [1].
* **Heavy Conflict Workloads.** The chosen MPLs and TRAN_SIZEs yield a relatively high mean number of deadlocks as a result of high levels of conflict (measured by mean number of waits). [2] indicates that a high deadlock rate is 100 per hour, and the simulation runs can exceed that figure. For example, run 4 resulted in approx. 300 deadlocks in one hour. These "non-negligible conflict levels.....facilitate understanding how the algorithms will perform under heavy conflict workloads, or when 'hot spots' exist in the database." [1].
* **Realistic levels.** A MPL of 5 is comparable to the general workload of a conventional order entry system [13].

5.3 Nature of the Test

Each of the 12 sets of data was processed 200 times (except for 3 runs where this was not possible because of the long simulation times involved). The sets of data were processed 200 times to produce sufficiently large sample sizes and so minimise the effects of outliers. Each set of data was processed both with EAGLE on (DDA) and then with EAGLE off (DLD). The transactions are passed to LOMAN from the Transaction module in a random, interleaved manner. LOMAN, after approving a lock event, passes the event sequence to EAGLE for approval. EAGLEs OBJECT or NO OBJECT response is then returned to LOMAN.

6. The Simulation results

There were 12 sets (runs) of transaction data constructed with various combinations of MPL, TRAN_SIZE and RESOURCE CONTENTION. Each of the 12 runs was executed (simulated) 200 (usually) times (cycles), with random selection of the sequence in which transactions were selected for forward progress. Three of the runs were not done for 200 cycles, because of the infeasible (long) simulation elapsed times involved. Each set of data was processed both with EAGLE on (DDA) and with EAGLE off (DLD). The number of locks granted, waits (denied locks), unlock requests, deadlock occurrences *etc.* were recorded in a report file. Results from the twelve simulation runs are summarised in Fig. 5 below. The first four rows (RUN NO., MPL, TRAN_SIZE, RESOURCE CONTENTION) describe the run. The SAMPLE SIZE is the number of times (cycles) the same set of transactions was processed. The last 4 rows contain the MEAN number of DEADLOCKS and the MEAN number of WAITS per cycle of transactions for both the EAGLE runs (DDA) and the runs without EAGLE (DLD).

RUN NO.	1	2	3	4	5	6	7	8	9	10	11	12
NO. TRANS (MPL)	2	2	2	3	5	3	4	4	4	3	5	4
NO. RESOURCES (TRAN_SIZE)	2	3	4	3	3	2	2	1-3	3	5	5	3
RESOURCE CONTENTION	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0	0.2
SAMPLE SIZE	200	200	200	200	40	200	200	200	110	200	50	200
MEAN WAITS (DLD)	1.56	2.13	2.84	5.82	20.2	4.36	10.8	3.08	12.6	8.23	24.3	6.23
MEAN WAITS (DDA)	1.76	1.93	2.35	6.05	19.4	4.56	11.0	2.55	11.9	7.83	25.3	6.05
MEAN DEADLOCKS (DLD)	0.52	0.73	0.87	1.65	6.73	1.19	3.64	0.52	3.91	2.33	7.04	1.36
MEAN DEADLOCKS (DDA)	0.57	0.58	0.61	1.65	6.23	1.23	3.62	0.33	3.63	2.09	7.4	1.26

Fig. 5. Results of EAGLE simulation runs

As an illustration, in Runs 2 and 3, MEAN WAITS using DDA is *less* than MEAN WAITS using DLD. In Runs 4 and 11, MEAN WAITS using DDA is *more* than MEAN WAITS using DLD. In Runs 2 and 3, MEAN DEADLOCKS occurring with DDA is less than that occurring with DLD. However, in Runs 6 and 11 the MEAN DEADLOCKS is more with DDA than when DLD is used.

A moving average provides a visual indication of the effect of introducing DDA into simulated locking activity. A moving average of the number of occurrences of deadlock per transaction cycle was calculated for each run. The moving average period was set to 20. The moving average chart for *one* of the Runs (Run 8) is shown in Fig. 6 below. The impact of EAGLE with DDA, relative to the "normal" Deadlock Detection and Resolution (DLD) without EAGLE, is clearly (visually) apparent. RUN 8 began with DLD processing and no "advice" or learning from EAGLE, with an average number of deadlocks per transaction cycle of 0.52. Then, beginning at transaction cycle 201, EAGLE began to "advise" LOMAN on the granting of locks, and to learn about deadlock-inducing sequences of lock activities. The average number of deadlocks quickly decreased to 0.33.

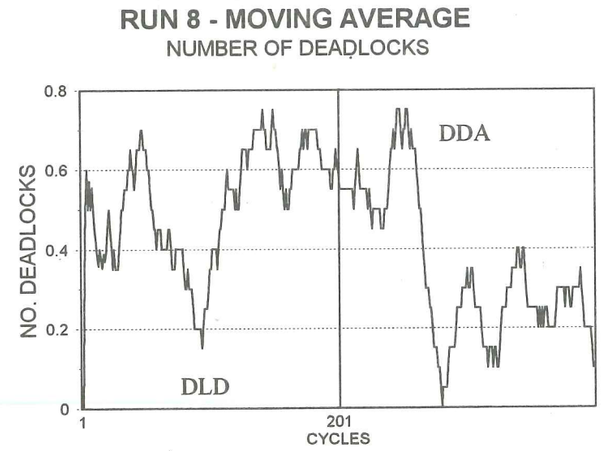


Fig. 6. Run 8: Moving average of deadlock occurrences without EAGLE, then with EAGLE.

However in Run 9, where the average number of deadlocks with deadlock detection was 3.91, the introduction of EAGLE does not seem to cause any noticeable perturbation in the number of deadlocks. After EAGLE is activated the number of deadlocks decreases marginally to 3.63. Figure 7 shows the moving average for Run 9.

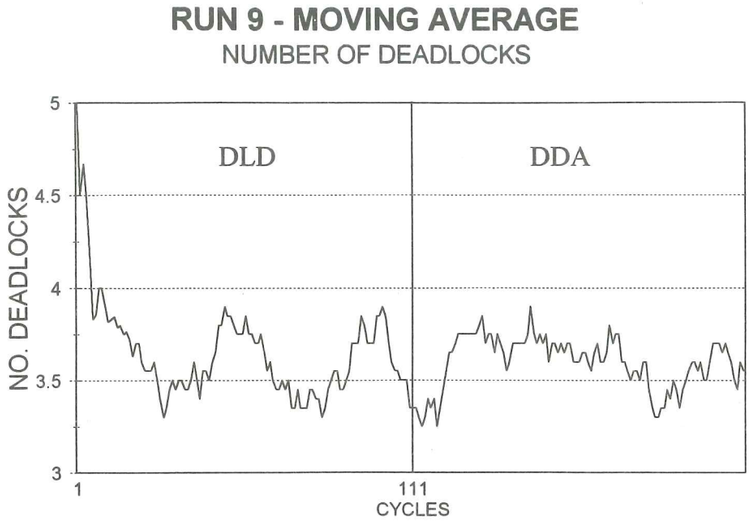


Fig. 7. Run 9: Moving average of deadlock occurrences without EAGLE and with EAGLE.

The simulation runs results indicate that, in certain situations, EAGLE reduces the number of deadlocks and waits occurring, while in other situations it seems to have no noticeable affect..

7. Evaluation of Results

Two sets of transactions were processed with the different deadlock treatment techniques (DDA and DLD). Other deadlock treatments exist [1], [3], [5], [10], [12], [13] and [22], but this study was restricted to a direct comparison between DDA and DLD. The effectiveness of DDA compared with DLD was measured with two of the "most important performance factors for locking" *viz*, "deadlock rate" (number of deadlocks) and "lock wait time" (number of waits). The third factor (lock conflict probability) is the primary measure of resource contention, which was not part of our investigation. All three factors "are interrelated" [13].

The null hypothesis was established as:

H_0: *There is no decrease in the criteria measures (number of deadlocks and number of waits) occurring with DDA compared with DLD.*

Alternate hypotheses were also tested, but are not reported in this paper. The Mann-Whitney U-Test was selected for (most of the) testing for significant difference [8]. Because of the nature of some runs, it was not meaningful to draw any statistical conclusion. For example, Run 1 was a 2 transaction, 2 resource simulation, so DDA could not avoid deadlock dynamically. All statistical tests were evaluated initially at a 95% level of confidence.

For example, Figure 8 shows the results of the Mann-Whitney U-Test for Run 8.


```
Comparison of Two Samples - Test for Number of Deadlocks
-------------------------------------------------------------------------
Sample 1: ESTAT8.NODL_DDA_
Sample 2: ESTAT8.NODL_DLD_

Test: Mann-Whitney

Average rank of first group = 181 based on 200 values.
Average rank of second group = 220 based on 200 values.
Large sample test statistic Z = 3.94227
One-tailed probability of equalling or exceeding Z; p = 0.00004037

NOTE:  400 total observations
```

Fig. 8. Comparison of Two Samples for Run 8

The average rank of the DDA sample for the number of deadlocks (181) is lower than the average rank of the DLD sample for the number of deadlocks (220). The computed test statistic Z, is 3.9422 and the p value is 4.04E-5. The null hypothesis can be rejected because the p value (4.04E-5) is lower than the set level of significance (0.05). It can be concluded that the number of deadlocks occurring under DDA is less than the number occurring under DLD.

In this run, at the 95% confidence interval the null hypothesis (H_0) was rejected.

However, for many of the runs (such as Run 9), the null hypothesis was not rejected.

Figure 9 presents a summary of the means and significance levels for each of the 12 runs.

Run	Wait Mean	DL Mean	DL Z stat	DL Sig level	Wait Z stat	Wait Sig level
1	1.56	0.53	-0.8025	0.2111	-0.8598	0.1949
2	2.13	0.74	3.1404	0.0008	2.5193	0.0059
3	2.84	0.87	5.9195	1.62e-09	4.4928	3.52e-06
4	5.82	1.65	-0.3498	0.6367	-1.9613	0.9751
5	20.15	6.73	1.5781	0.0573	1.2622	0.1034
6	4.36	1.19	0.0981	0.4609	-0.5423	0.7062
7	10.8	3.64	0.0878	0.465	-0.3374	0.6321
8	3.08	0.52	3.9423	4.04e-05	4.0256	2.84e-05
9	12.65	3.91	-0.6492	0.7419	-0.7406	0.7705
10	8.23	2.33	1.7812	0.0374	0.4027	0.3436
11	24.32	7.04	-0.9991	0.8411	-1.2176	0.8883
12	6.23	1.36	1.211	0.1129	0.9812	0.1632

Fig. 9. - Summary of Results

The first column is the run number, the second and third columns are the wait and deadlock means, and the last four columns show the deadlock and wait Z statistic and significance levels, respectively.

Out of the 12 test situations, H_0 was rejected in 4 of the runs and accepted in 8 runs. The shaded area shown in Fig. 10 shows the 4 runs where there was a significant decrease in the number of deadlocks.

Run	DL Mean	DL Z stat	DL Sig level
3	0.87	5.9195	1.62e-09
8	0.52	3.9423	4.04e-05
2	0.74	3.1404	0.0008
10	2.33	1.7812	0.0374
5	6.73	1.5781	0.0573
12	1.36	1.211	0.1129
6	1.19	0.0981	0.4609
7	3.64	0.0878	0.465
4	1.65	-0.3498	0.6367
9	3.91	-0.6492	0.7419
11	7.04	-0.9991	0.8411

Fig. 10. Test results in deadlock significance order

The shaded area in Fig. 11 shows the 3 runs where there was a significant decrease in the number of waits.

Run	Wait Mean	Wait Z stat	Wait Sig level
3	2.84	4.4928	3.52e-06
8	3.08	4.0256	2.84e-05
2	2.13	2.5193	0.0059
10	8.23	0.4027	0.3436
5	20.15	1.2622	0.1034
12	6.23	0.9812	0.1632
6	4.36	-0.5423	0.7062
7	10.8	-0.3374	0.6321
4	5.82	-1.9613	0.9751
9	12.65	-0.7406	0.7705
11	24.32	-1.2176	0.8883

Fig. 11. Test results in wait significance order

The combined results show that only in runs 2, 3 and 8 there is both a significant decrease in the number of deadlocks and the number of waits after EAGLE is introduced. Further examination of the results indicates that DDA was not associated with a significant decrease in the number of deadlocks when MEAN DEADLOCKS was greater than 1, and, was not associated with a significant decrease in the number of waits when MEAN WAITS was above 3.08. Runs 2, 3 and 8 all have a significantly decreased number of deadlocks and waits associated with the use of DDA. The number of deadlocks is significantly decreased with confidence intervals in excess of 99.998%, and the number of waits is significantly decreased with confidence intervals in excess of 99.994%. DDA reduces the number of deadlocks and waits, compared with DLD, in low conflict situations, typified by a mean number of deadlocks below 1 and a mean number of waits below 3.08. See [8] for a fuller discussion of the simulation run results for various combinations of transactions, parameters and resources.

There are two possible reasons why DDA is effective in low conflict situations (characterised by a mean number of deadlocks below 1);

1. Multiple deadlock situations imply that there are multiple event sequences that can give rise to deadlock. Each of these event sequences is represented as a script in the script base. The larger the number of scripts, the higher the likelihood of a match occurring between an observed event sequence and a stored script, and hence the higher the likelihood of an OBJECT. An OBJECT ceiling is set to limit the number of sequential OBJECTs occurring.

It could therefore be possible that EAGLE becomes less effective in high conflict situations. Too many situations are identified where deadlock seems imminent, which results in multiple OBJECTs and consequently multiple ceilings. The result is that EAGLE is unable to effectively decrease deadlocks.

2. Another possible reason why EAGLE is ineffective when the number of deadlocks is more than 1, may be linked to the number of script patterns required to represent the multiple deadlock sequences. In high conflict situations more combinations of transaction processing can result in deadlock, and multiple deadlocks can occur within the processing of a single batch of transactions. So the number of scripts required to represent the multiple deadlock situations increases, as does the likelihood of an unfolding event sequence not matching any of the stored scripts.

In order to determine whether either of these factors affects the performance of DDA, more tests need to be run where parameters such as the ceiling level, activation level, and clean level are monitored.

8. Conclusion

Script-based knowledge representation has proved to be appropriate for treatment of potential deadlock situations based on learning from previous experience. EAGLE (DDA) significantly decreases the number of deadlocks and the number of waits in low conflict situations. However, DDA (currently) does not significantly decrease the number of deadlocks or the number of waits in high conflict situations. Further work is needed in dealing with high conflict situations, especially in terms of the matching techniques that are used. The higher the conflict situation, the more complex the matching process becomes. Future work needs to consider the use of other "expert" techniques within the lock manager, such as neural networks.

References

[1]Agrawal R, Carey M J and McVoy L W, (1987), The Performance of Alternative Strategies for Dealing with Deadlock in Database Management Systems, *IEEE Transactions on Software Engineering*, Vol. SE-13, **12**, pp. 1348-1363.

[2]Bachman, C.W. (1973). The Programmer as a Navigator, in *Communications of the ACM*, Vol. 16, No. 11, November, pp. 653-658.

[3]Belik, F. (1990) An Efficient Deadlock Avoidance Technique. *IEEE Transactions on Computers*, Vol. 39, No. 7, July, 1990, pp. 882-888.

[4]Benoit, J.W., Davidson, J.R., Hofman, E.J., Laskowski, S.J. and Leighton, R.R. (1987) Integrating plans and scripts : an expert system for plan recognition, in *Proceedings of the Third Annual Expert Systems in Government Conference*, IEEE Computer Society Press, Washington, pp. 201-207.

[5]Bernstein, P.A., Hadzilacos, V. and Goodman, N. (1987). *Concurrency Control and Recovery in Database Systems*, Addison-Wesley Publishers Ltd., Reading, Massachusetts.

[6]Blewett, C. N. and Erwin, G.J. (1995) The Application of Scripts to Deadlock Avoidance, in *Proceedings of SAICSIT95, Research & Development Symposium*, May, Pretoria, South Africa, pp. 33-47.

[7]Blewett, C. N. and Erwin, G.J. (1996) EAGLE: A script-based embedded expert system for deadlock avoidance, paper accepted for *Proceedings of Third World Congress of Expert Systems* (WCES-3), 5-9 February, Seoul, South Korea.

[8]Blewett, C.N. (1996) A Script-based Dynamic Deadlock Avoidance Technique, *M. Comm. Thesis*, Business Information Systems Section, Faculty of Economics and Management, University of Natal, Durban, South Africa.

[9]Chen, D.C. (1985) Progress in knowledge-based flight monitoring, in Proceedings of the Second Conference on Artificial Intelligence Applications, *The Engineering of Knowledge-Based Systems*, IEEE Computer Society Press, Washington, December, pp. 441-446.

[10]Date, C.J. (1990). *An Introduction to Database Systems*, Volume I, Fifth Edition, Addison-Wesley Publishing Coy., USA, 1990.

[11]Erwin, G.J., and Bowen, P.A.(1994). Describing the building procurement process using script-based knowledge representation. *South African Journal of Science*, Vol. 90, No. 10, October, 1994, pp. 543-546.

[12]Everest, G.C. (1986). *Database Management : Objectives, System Functions and Administration*, McGraw-Hill, USA.

[13]Jenq, B., Twichell, B. and Keller, T. (1989). Locking Performance in a Shared Nothing Parallel Database Machine, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 1, No. 4, USA.

[14]Larsson, J.E. and Persson, P. (1986) Knowledge representation by scripts in an expert interface, in *Proceedings of the American Control Conference*, pp. 1159-1162.

[15]Laskowski, S.J. and Hoffman, C.N. (1987) Script-based reasoning for situation monitoring, in *Proceedings of AAAI-87,* July, pp. 819-823.
[16]Markosian, L.Z., Rockmore, A.J., Keller, K.J. and Gaan, M.R. (1985) An expert system for air-to-air combat management, in *Proceedings of the Eighteenth Asimolar Conference on Circuits, Systems, and Computers*, pp. 112-116.
[17]Mutch1er, C.N. and Laskowski, S.J. (1991) Script matching and belief propagation, in *Proceedings of the 5th IEEE International Symposium on "Intelligent Control"*, September, MITRE Corporation.
[18]Schank, R.C. and Abelson, R.P. (1977) *Scripts, Plans, Goals and Understanding : An Inquiry into Human Knowledge Structures*. Lawrence Erlbaum Associates, Hillsdale, New Jersey.
[19]Schank, R.C. and Riesbeck, C.K. (Eds.) (1981) *Inside Computer Understanding : Five programs plus miniatures*. Lawrence Erlbaum Associates, Hillsdale, New Jersey.
[20]Schank, R.C. and Childers, P. (1984) *The Cognitive Computer : On language, learning and artificial intelligence*. Addison-Wesley, Reading, Massachusetts.
[21]Waterman, D.A. (1986) *A Guide to Expert Systems*. Addison-Wesley, Reading, Massachusetts.
[22]Wiederhold, G.W. (1983) *Database Design*, 2nd. Edition. McGraw-Hill Book Company, New York.

Acknowledgements

Support work for EAGLE was done by Karl Fischer, a Business Information Systems (Honours) student in the Department of Accounting and Finance, Business Information Systems Section, University of Natal, King George V Ave., Durban, South Africa, 4001. LISP code for the Matcher/Analyzer was developed by Anna Richter c/o Mr A. Richter, Dept. of Mechanical Engineering, University of Natal, King George V Ave., Durban, South Africa, 4001.

PARALLELISM: AN EFFECTIVE GENETIC PROGRAMMING IMPLEMENTATION ON LOW-POWERED MATHEMATICA WORKSTATIONS

H. Suleman, M. Hajek
Department of Computer Science
University of Durban-Westville
Private Bag X54001
Durban 4000
hsuleman@pixie.udw.ac.za
mhajek@pixie.udw.ac.za

Abstract

Mathematica has proven itself to be a suitable platform on which to develop prototype Genetic Programming applications. However, due to the sheer complexity of genetic programming calculations, non-trivial problems cannot be solved on a single Mathematica workstation. A distributed algorithm is suggested to eliminate such restrictions on the problem domain. A client-server network is utilised to model a system of multiple populations under simultaneous evolution. Regular migration of members, through the medium of the server, results in suitable genetic material being filtered through to other populations. This multi-population model is contrasted with the single-population standard approach in terms of its performance and utility value.

Introduction

Genetic Programming is a generalisation of the standard genetic algorithm. It falls under the broad category of Evolutionary Programming, including both crossover and mutation in its genetic operations set.
Mathematica is proposed as an alternative language of implementation for prototyping Genetic Programming (GP) algorithms. Historically, a large number of GP implementations are done in LISP, with a fewer number in other 3GL languages for greater speed and better data management. Mathematica is an ideal option because of the manner in which data and code can be freely interchanged; this being the case in LISP as well. Since it is essentially a mathematical modelling language, it performs well in problem domains that are mathematically orientated. Such an implementation can also be optimised to take advantage of the specific features of Mathematica, especially its multi-paradigm approach to programming, encompassing all of procedural, declarative and functional techniques.

Evaluation of Implementation

Any implementation of a GP, but especially one in Mathematica, is likely to encounter the problem of excessive execution times. For a non-trivial problem, the parameters cannot be biased towards a known solution, making the lengthy calculations unavoidable.
A time-analysis of a typical execution of the algorithm indicates that the time taken to perform the genetic operations is generally proportionate to the size of the population. However, the time taken to evaluate fitnesses increases as the complexity of the expressions increases. Since the fitness calculations take up a large portion of the time, it may help if these were done in parallel on multiple processors.
Another important factor to be considered is the level of variety in the population. Initially, the size of population is chosen so that sufficient variety is seeded into the first generation. However, as crossover proceeds, many characteristics are lost to subsequent generations. Mutation is introduced to recover these lost attributes. However, mutation is not an ideal solution since it seeks to restore lost genetic material, rather than protect existing potential solutions. Running a GP several times can produce highly contrasting results since the genes considered are different in each run. Thus, repeating a GP run several times is one possible way of ensuring that the final solution found is as near as possible to

the optimal solution. Another option would be to split a large population into sub-populations and execute the GP on these.
Many problem domains have only highly complex solutions. In order to generate such solutions using a GP, a large amount of memory is required. A typical problem to generate a parser for a high-level language requires that the function set spanning a complete language be used as an input parameter. To adequately cater for this variety, the population of solutions needs to be very large. Such problems demand either stringent hardware configurations or innovative approaches to memory management. One innovation would be to split up the population and process each segment separately in sequence, or in parallel if multiple processors are available.
It is clear from the above discussion that many problems and shortcomings of the standard GP can be overcome by using multiple populations evaluated in parallel on multiple processors.

Parallel Processing Model

Many models exist to transform the standard genetic programming algorithm into a parallel one. Some models consider the population to be singular and break up the work to be done at the lower level of the individual operations or calculations. This is obviously more suited where a parallel or distributed operating system is available. In other cases, the population needs to be split up into sub-populations to facilitate parallelism, as was done for this study.
The most important consideration after dividing the population is whether to utilise discrete or continuous population evaluations. Continuous evaluations provide a better model of reality but incur the added expense of greater inter-population communication. Discrete evaluations are an attempt to cut down on inter-population communication.
Where a single population was evaluated for the standard GP, multiple populations will have to be evaluated for the parallel algorithm. These populations each evolve through one generation in a single cycle of the algorithm. Thereafter, a global solution can be sought, and a new cycle may begin.
Just searching for global solutions among the various sub-populations would reduce the problem to a multiple-run equivalent. Instead a mechanism to bind the sub-populations together must be introduced. This would imply members of one sub-population interacting with members of another sub-population through either reproduction or crossover. This solution has the disadvantage that genetic operations have to be performed on multiple sub-populations simultaneously, thereby reducing the ability to process the populations in parallel.
Migration of members is an attempt to achieve both these goals without the need for distinct operations on the data. Analogous to migration of people, members of a population may migrate from one population to another. This migration is fitness-proportionate to ensure that the important genetic material of a sub-population has a higher probability to migrate. Since migration does not involve any genetic operations, it can be done in-between the discrete evolution of populations.
Migration introduces the added problem of selecting criteria for the movement of members. If all members were allowed to migrate to any sub-population, the problem of losing genetic material would not be solved, as sub-populations with very fit individuals would very soon influence all other sub-populations. In order to preserve the genetic material of any sub-population, migration must be restricted to a subset of the set of sub-populations.
To restrict migration, it is useful to map the set of sub-populations onto a mathematical surface, introducing a specific spatial distribution into the algorithm's parameters. If all the sub-populations were arranged in a straight line, each sub-population would only allow migration with its immediate neighbours on either side. Restrictions could also be set to dictate the probability with which migration will occur between further sub-populations. In this study, migrations were limited to immediate neighbours to reduce the inter-population communication during a migration operation.
If the surface used was a circle, migration would also be possible between the first and last sub-populations, which would result in all sub-populations having similar behaviour. Such linear arrangements will result in clusters of sub-populations with common sub-expressions occurring at various points along the surface.
Generalisations of this technique may involve adding more dimensions to the surface. The experiments in this study used a two-dimensional surface (Figure 1). Sub-populations were distributed to a square grid on the surface, with those at the ends and corners wrapping around to the opposite end. Thus, each sub-population can effectively allow migration between itself and eight other sub-populations. This would give rise to regions with common characteristics in the plane of the surface.
This entire sub-population system is analogous to reality as it depicts the manner in which people live in separate communities and then move among these communities.

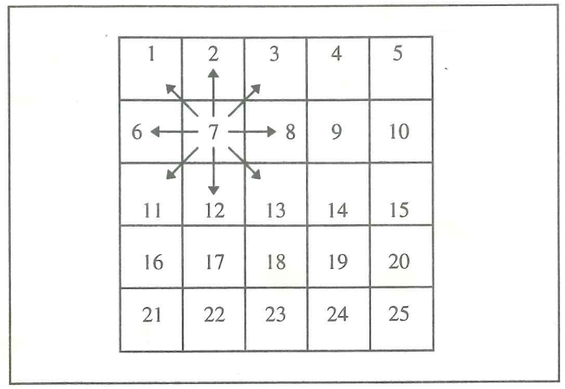


Figure 1: Rectangular spatial distribution of sub-populations showing migration possibilities for sub-population 7

Distributed Implementation

The parallel GP algorithm, as outlined above, was implemented in Mathematica for MS-DOS using a Windows for WorkGroups Network Operating System (NOS). Mathematica was retained as the language of choice since it allowed painless modelling of new problems. The NOS chosen was used simply because of availability and not because of any assumed superiority.
Instead of using programs that communicate directly over the network, precedence was given to code that communicated using primitive file-based data exchanges. Since all communication was through the opening, closing and deleting of files, the system would be independent of any particular NOS.
A common dilemma when splitting up the population, is whether the population data should reside on a single computer or on multiple computers. If the population data resides on a single computer, there will be extra communication along the network. If the population data resides on multiple machines, those machines will dictate the number of sub-populations and those machines will have to be present for the full duration of the calculations to prevent drastic scenario changes. The single machine option was chosen to keep the number of processors flexible throughout the lifespan of the calculations. All processors would mount the shared directory on the "server" in order to use the common population store.
Many initial attempts at parallelizing a single-computer GP were abandoned due to problems with the particular platform chosen. These techniques may nonetheless be used in future implementations.
The first algorithm used file sharing and locking to ensure that two processors did not attempt to process the same population. Initially, a set of dummy "flag" files were created, one corresponding to each population. Processors would then lock a flag file, process the relevant population and delete the flag file. The file locking would pre-empt the possibility of any other processor corrupting or duplicating the calculation. After all the flag files were deleted, one of the processors would perform migration, using a migration flag, and then generate a set of new flag files. Each processor selected the population it would process by searching through the flag files for one that wasn't locked; the process was dynamic and depended solely on the processors. This method did not work because the OS locking mechanism failed when heavily loaded.
As an alternative, deleting files and searching for files was used instead of file locking. This also failed since the indivisibility of such operations was not guaranteed.
Eventually, it was decided that the processors could not themselves decide which populations to process. A static scheduler was introduced to submit jobs to the individual processors. This scheduler would then take over the role of creating the flag files in pre-specified directories. The individual processors would search their specific directories for flags. When one was found they would process the relevant population and then delete the flag file. The scheduler would constantly check for such deletions and immediately schedule another job to that processor. After all the population evolution is done, the scheduler submits a "Migrate" job to one of the processors and thereafter begins the cycle over again.

Analysis of Implementation

A series of tests was run on the parallel Mathematica implementation to determine its effectiveness compared to the standard single-population approach. All tests were run on a network of 486-DX33 computers with one acting as the server and the other being clients. The server contained the population data and ran the scheduler while the clients ran the various Mathematica population processing programs. The parameters (Table 1) for all calculations was identical except for a variance in the number of clients used.

Table 1 : GP Parameters

Parameter	Value
Crossover Probability	90%
Mutation Probability	10%
No Of Sub-populations	9
Total Population Size	450
Minimum Solution Fitness	100%
Maximum Initial Size	5
Maximum Size	17
Maximum Complexity	50
Migration Probability	25%
Migration Percentage	10%
Migration Std. Deviation	5%

"Total Population Size" refers to the sum of the sizes of the sub-populations. "Maximum Initial Size" is the maximum depth that an expression can take on at generation 0. "Maximum Size" is the maximum depth that an expression can reach as a result of genetic operations. "Maximum Complexity" is the maximum number of nodes that an expression may contain. "Migration Probability" is the probability with which migration occurs after between two sub-populations. "Migration Percentage" is the average number of individuals that are exchanged during a migration operation. "Migration Std. Deviation" controls the amount by which the Migration Percentage may vary.

The problem was a simple symbolic regression one, where a curve of unspecified form was to be fitted through a set of 10 sample points. The equation used to generate the sample data was :

$$f(x) = x^4 + x^3 + x^2 + x$$

The function set used was {+(3), +(2), x(3), x(2), -(1), /(2)} with the associated arity in brackets, to speed up convergence. Using only functions with arity 2 in prior experiments, the algorithm was shown to converge but at a much slower rate. The terminal set contained only the unknown variable.

Every run of the algorithm produced a perfect solution. The rather large population size, relative to this problem, was chosen so that the efficiency of the algorithm could be measured instead of the ability to find a solution; previous experiments on a single population had confirmed that GP could resolve this problem for smaller populations.

The time taken was as follows:

Table 2 : Benchmark times

	sec.	ave. secs. /gen.	min.	max.
1 client	7365	484	415	494
3 client	2514	220	190	254
9 client	2489	188	149	209

Each experiment was repeated five times and a mean of these was calculated. The second column refers to the average time taken for the algorithm to reach the perfect solution. The third column is the average time taken to produce a new generation by evolution, followed by the minimum and maximum times to produce a new generation.

The experiment was also repeated using a single population to check the net effect of using the parallel approach.

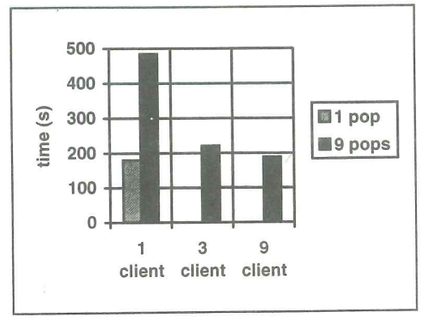


Figure 2 : Graph of generation calculation times (in seconds) for each scenario

It is apparent that the time taken to generate a new full population decreases as the number of clients increases. However, this increase is not proportional to the increase rate of the number of clients. This is due to the fact that the system is affected by many other factors than the number of clients.

As the number of clients increase, so does the load on the server and the amount of network communication. Thus, more clients do not necessarily increase the efficiency of the network.

Migration was done on a single computer. Thus the entire system would come to a halt while the migration operation was being carried out. The rationale behind this was that migration would take far less time than evaluation of fitnesses so the latter had greater priority to be distributed. However, when a large number of clients were used, the time taken to generate new populations became less than that to migrate. Thus, migration affected the algorithm more as the number of processors was increased.

Since Mathematica eventually ran out of memory when doing long sequences of calculations, it was restarted before each population evolution cycle, adding to the overhead of the algorithm.

When compared to the single-population model, all versions of the multi-population still take longer to find the solution. However, these are all attributable to the specifics of the implementation. As a result of parallelism, the algorithm can now be applied to problems of arbitrary size and complexity, without fear of overrunning the resources of the computer. Also, one iteration of the algorithm would now find solutions that previously required multiple runs to maintain variety of genetic material.

Conclusion

Genetic Programming is a powerful technique to solve problems with no explicit solution methodology. Two of its shortcomings are execution time and loss of genetic material, which can be addressed by using a parallel algorithm. Parallel algorithms, although incurring the overheads of inter-process communications, allow large problems to be solved irrespective of the size and complexity of the solution or the computational power of the machines.

The particular implementation on Mathematica can be improved by parallelizing the migration operation. In addition, the server/network can be changed to enable faster communication. With just these two improvements, the parallel algorithm will definitely surpass the standard approach in even the category of evaluation time.

References

1. Koza, J.R., *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.
2. Nachbar, R. B., *The Mathematica Journal* 5(3), p36-47 Genetic Programming, Miller Freeman Publications, 1995.
3. Freeman, A., *Simulating Neural Networks with Mathematica*, p271-273, Addison-Wesley, 1994.

FEATURE EXTRACTION IN NEURAL NETWORKS: AN APPLICATION IN HANDWRITTEN CHARACTER RECOGNITION

Deshendran Moodley
Computer Management (Pty) Ltd
P O Box 451, Umhlanga Rocks, 4320

Vevek Ram
Department of Computer Science and
Information Systems, University of Natal,
Private Bag X01, Scottsville, 3209

Abstract

This paper examines and provides an overview of feature extraction techniques that are currently used in neural networks for image recognition. These include moment invariants, Zernike moments, Fourier descriptor techniques, Gabor and wavelet filters and the Neocognitron. An implementation of an handwritten character recognition system is discussed to illustrate the practical significance of feature extraction. The methods of Zernike moments and the Neocognitron are used for feature extraction and the multilayered perceptron is used as the classifier. The results are also compared to the same application without feature extraction.

Introduction

Shape and pattern recognition is an essential function of computer-based vision systems used in industrial automation. Artificial Neural Networks have played a prominent role in this area and many diverse manufacturing applications such as product inspection and packaging, robotics and remote sensing have been successfully developed. In a typical application, a feature vector which is a numerical representation of an image, is presented to an ANN for classification. In most cases, the overwhelming size of the feature vectors require networks which are computationally impractical. Feature extraction is a process which drastically reduces the size of the feature vector when compared to the original image, by reducing redundancy and retaining only the information necessary for discrimination. Many techniques for achieving feature extraction exist and this paper examines and provides an overview of those that are currently used.

The multilayered perceptron (MLP) (Rumelhart, Hinton & Williams, 1986), a popular neural network architecture, consists of three layers, the input layer, the hidden layer and the output layer with each layer containing processing elements, called neurons, which are connected to the neurons in the previous layer. The classification process involves the presentation of a feature vector to the input layer and the producing of a response from a specific neuron in the output layer, corresponding to an output class. Initially a set of prototype pairs of feature vectors and desired responses are presented repetitively to the network. After each presentation, differences between the actual output and the desired output of the network are used to modify the strength of connections within the network. Eventually the network reaches a state where these differences are negligible and the network is ready to perform classification. For digital image recognition the digital image forms the feature vector and the output classes are all the possible classification categories for the image. The use of the multilayered perceptron in digital image recognition has two limitations. Firstly, the unnecessary processing of redundant information not needed for discrimination between output classes. For example, if the task was to classify fruit the information about the size, position or how the fruit is rotated in the image is of no significance. Secondly, classification of large digital images necessitates a large number of neurons in the input layer, which leads to an increase in the computational resources needed for the operation of the network and the time needed for training the network.

The above limitations of the MLP can be overcome by the introduction of a feature extraction stage in which the digital image is processed so that only relevant properties of the image required for discrimination between the output classes are extracted. This new feature set will then form the input

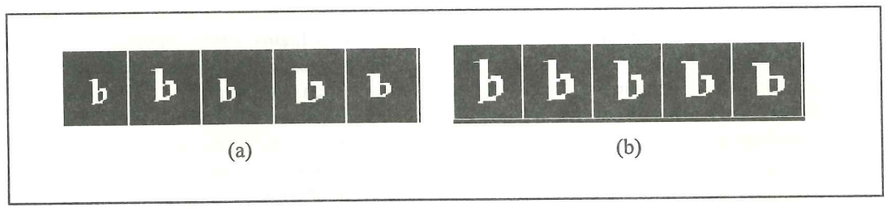
to the neural network. In this way redundant information will be removed and the size of the feature set reduced. Generally feature extraction methods must have the following properties (Khotanzad & Lu, 1991). Extracted features must retain much of the discriminant information present in the original data. Features should have small intra-class variance, that is, slightly different shapes with similar general characteristics should have similar numerical values. Features must also have large inter-class differences, that is, features from different classes should be quite different numerically.

Feature extraction methods

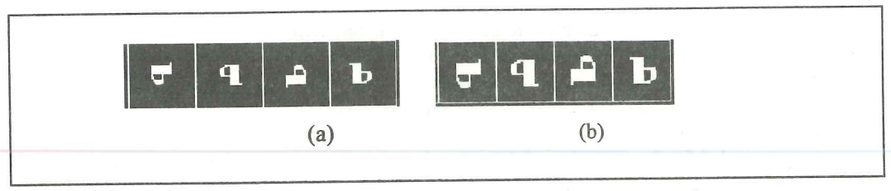
The shape of objects in the image and variations in texture within the image are especially important in digital image recognition. Different textures in images can be distinguished by their preferred direction or orientation in the image and by their spatial frequency, that is, how fine (more detail) or coarse (less detail) the pattern is. The detection of variations in texture helps in the segmentation of an object and its background in the image and the identification of the shape of the object. The feature extraction methods discussed below can be classified by these properties of the image which they capture. Moment invariants (Jain, 1989), Zernike moments (Khotanzad and Hong, 1990), Fourier descriptors (Kulkarni, 1994) and the Neocognitron (Fukushima, 1988) capture the shape of an object in the image while Gabor filters (Daugman, 1988) and wavelets (Mallat, 1989) help to distinguish between textures within an image. Since Zernike moments and the Neocognitron are selected for the experiments, they are discussed in detail. The other methods are included for the sake of completeness.

Zernike moments

Zernike moments (Khotanzad and Hong, 1990) have been used previously for shape recognition tasks. The representation consists of 47 positional, size and orientation invariant features which are the magnitudes of the complex valued Zernike moments of a digital image. The original image can be reconstructed from these moments, thus giving an indication of the quality of the representation. Firstly, the image is standardised with respect to size and position. Thereafter the rotational invariant Zernike moments of these standardised images are calculated. Figure 1 shows five instances of the letter 'b' with variations in size and position before and after standardisation and figure 2 shows four instances of the letter 'b' with variations in orientation before and after standardisation. The first four Zernike moments of the images in figure 2b are shown in table 1.



1. Five instances of the character b (a) before and (b) after standardization



2. The letter b rotated through angles of 90° , 180° ,270°

Table 1 The magnitudes of 4 Zernike moments for the rotated images in figure 2

	90°	180°	270°	0°
|A[2,0]|	207.935751	207.935751	207.935751	207.935751
|A[2,2]|	3.246915	3.246915	3.246915	3.246915
|A[3,1]|	52.198420	52.198420	52.198419	52.198419
|A[3,3]|	2.201566	2.201566	2.201566	2.201566

The Neocognitron

The Neocognitron (Fukushima, 1988), is a multilayered neural network architecture, based on the feed forward architectures in biological systems, which combines the feature extraction and classification stage. The Neocognitron is insensitive to variations in position and distortions in the input image. This architecture has been used predominantly for shape recognition tasks. The Neocognitron incorporates the feature extraction stage within the network and this in itself is both convenient and important. The introduction of a separate feature extraction stage, for example Zernike moments results in uncertainty about the degree of processing carried out by the neural network as the feature extraction stage may tend to oversimplify the classification process. The Neocognitron is insensitive to scale, changes in position and even distortions of objects in the input pattern. The unsupervised training used for the Neocognitron resembles closely the processes in biological visual systems and the system determines the features to be detected. Even though this training method does not produce good results in practice, it does give insight into the operation of biological visual systems. To produce better results, but moving away slightly from the biological paradigm, supervised training methods have been developed. These are more suited to pattern recognition and allow the teacher to control which features are to be detected.

Moments invariants

Moment invariants are derived from the central moments, m_{pq} , of an image (Jain, 1989)

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y)$$

These moments provide a positional, size and orientation invariant representation of an image. They have been used for shape recognition and provide a representation consisting of seven features. A significant disadvantage is that there is no measure of how good a representation of the original image these moments provide.

Fourier descriptors

Fourier descriptors provide a positional, size and orientation invariant representation of a digital image. This method has been investigated for 2-D aircraft recognition (Kulkarni, 1994). New ways in which to use Fourier descriptors for image representation are currently being investigated (Kauppinen & Seppanen,1995).

Gabor filters

Gabor filters (Daugman, 1988) operate similarly to structures in biological systems, that is, they categorise areas of an image into different ranges of orientations and spatial frequencies. It is one of the few representations that can simultaneously distinguish between orientations and spatial frequencies which is important for texture classification. However the calculation of the Gabor filters for a digital image is difficult.

Wavelets

The wavelet representation (Mallat, 1989) uses the differences between images at different resolutions to represent a digital image. This representation also, but to a lesser extent, is able to simultaneously distinguish areas of the image according to different orientations and spatial frequencies. Calculation of this representation is less difficult than for Gabor filters.

The application of handwritten digit recognition was chosen to practically illustrate the role of feature extraction in image recognition. Two feature extraction models, Zernike moments and the Neocognitron using supervised training, were chosen as suitable for two dimensional shape recognition. Three experimental applications were designed and implemented to demonstrate the significance of introducing a feature extraction stage into the handwritten digit recognition system:

- Experiment I consisted of a system with no feature extraction.
- Experiment II consisted of a system with a feature extraction separate feature extraction stage using Zernike moments.
- Experiment III consisted of a system which combined the feature extraction and the classification stages in the form of the Neocognitron.

These experiments were tested on data obtained from the National Institute of Standards and Technology (NIST) in the USA. The data consisted of isolated handwritten digits. Even though in some instances the characters were cut off or parts of two characters were found in some images (segmentation error), the segmentation process used to produce the isolated characters was considered overall to be quite good since:

- the characters had minimal variations in size.
- the characters had little or no variations in position as all the characters filled the entire image.

A selection of these characters were rotated and a new data set, data set II, was constructed which contained a combination of the rotated and the normal characters. The experiments were also tested on data set II. The results of these experiments are displayed in table 1 and table 2.

Table 2 Performance of the three experiments on data set I.

Experimental application	Accuracy rate on training set	Accuracy rate on test set
I. MLP	100%	83.8%
II. Zernike moments	55-65%	50 - 60%
III. Neocognitron	-	41%

Table 3 Performance of the three experiments on data set II.

Experimental application	Accuracy rate on training set	Accuracy rate on test set
I. MLP	100%	47%
II. Zernike moments	100%	67%
III. Neocognitron	-	28%

The results of these experiments can be summarised as follows:

- The introduction of a feature extraction method in the image recognition system, developed in experiment II, reduces the size of the feature vector and thus the amount of resources needed for implementing, training and testing of the artificial neural network. This is especially noticeable in the number of neurons needed for the input layer.
- The feature extraction process allows the system to be invariant to certain irrelevant properties of the input data.
- The disadvantages of introducing a feature extraction stage into the system, is the additional processing that is required and the loss of information that occurs when reducing the input data. This must be weighed against the advantages of invariancy and reduction in size of the feature vector which feature extraction provides. It can thus be concluded that if there are negligible variances in parameters such as size and position, then the only advantage of the feature extraction stage is reduction in size of the feature vector.

Conclusions

An artificial neural network, the multilayered perceptron, has been investigated for digital image recognition applications. The limitations of the multilayered perceptron have been addressed by the introduction of a feature extraction stage. An experiment in the recognition of handwritten characters demonstrated the practical effectiveness of two particular methods namely Zernike moments and the Neocognitron. There are many other applications where feature extraction can certainly play a major role in contributing to the effectiveness of recognition systems, and it is hoped that this research has provided some insight in order to make better decisions regarding their design and development.

References

Daugman, J. G. (1988), Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol 36 no 7, pp. 1169- 1179.

Fukushima, K. (1988), A Neural Network for Visual Pattern Recognition, *IEEE Computer*, Vol 21, no. 3, pp. 65- 74.

Jain, A. K. (1989), *Fundamentals of digital Image Proessing*, Prentice-Hall, United States of America.

Kauppinen, H., Seppanen, T. Pietikainen, (1995), An Experimental Comparison of Autoregressive and Fourier-Based Descriptors in 2D shape Classification, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 12, no 2, pp. 201- 206.

Khotanzad A., and Lu J. -H. (1991), Shape and Texture Recognition by a Neural Network, in *Artificial Neural Networks and Statistical Pattern Recognition*, edited by Sethi I. K. & Jain A.K. , Vol. II, pp. 109-131, Elsevier Publishers B. V, Amsterdam.

Khotanzad, A., Hong, Y. H. (1990), Invariant Image Recognition by Zernike Moments, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, no. 5, pp. 489-498.

Kulkarni, A. D. (1994), *Artificial Neural Networks for Image Understanding*, Van Nostrand Reinhold, New York.

Rumelhart, D. E., Hinton, G. E. , Williams, R. J. (1986), Learning Internal Representations by Error Propagation, In *Parallel Distributed Processing - Explorations in the Microstructure of Cognition, Volume 1: Foundations*, edited by Rumelhart, D. E., McClelland, J. L.and the PDP Research Group, pp. 318-362, MIT Press, United States of America.

REAL-TIME CONTROL THE SYSTEM MODEL - AN HOLISTIC APPROACH TO SYSTEM DESIGN

TN CONSIDINE

APRON SERVICES (PTY) LTD
JOHANNESBURG INTERNATIONAL AIRPORT
PO KEMPTON PARK
1600
(011) 978-3493

The Real-time Control System Model is a system design methodology which provides robust systems based on Object technology. The paper will detail the components of the model including the control layers and the time-based levels into which the model is divided. The application of the Model in the design of an operational system is described in detail particularly the sensor, user interface and world model layers.

In the United States 35% of all software projects are cancelled before completion[3]. 91% of large projects fail as do 84% of small projects[4]. Indeed a snap survey of 14 magazines published in the past year showed 16 articles published on Software Quality or the lack of it. I do not know how South Africa compares to the American situation but I suspect that it is much the same.

Why does software fail?

A recent study by Software Productivity Research showed that the most significant factors were the lack of automated software cost estimating tools, automated software project management tools, effective quality control and effective tracking of software development milestones[5]. All of these are needed because too much software is written from scratch.

It seems logical to assume use of standard modules or objects in systems can promote successful development. However, the use of standard software modules seems, in many cases, to be beyond the reach of developers. Why should this be? Taking an extreme view of software development it could be said that software development is similar to building a house. To build a house you simply buy the objects, bricks, window frames, baths and so on, assemble them and you have a house. This does not work properly since there are choices in housebuilding, floor finishes, for example. So a house is not simply a collection of objects but more accurately, a collection of standard objects, customisable objects and new objects. Over many thousands of years the building industry has standardised on customisation of objects. Cost features in the equation as well, the more expensive the house, the less the use of standard objects and the more the use customised or new components.

Use of standard modules in house building has it's place but is subject to the factors outlined before. Similarly, use or re-use of components in a system is a blend of standard components, customisable components and new components. It follows then, that there is a hierarchy of re-usability which depends on the system under consideration. Or to put it another way, if you were creating special effects for the Jurrasic Park versus Godzilla film you would use mostly new components whereas if you were doing an accounting system for your hairdresser you would use standard components almost everywhere. Re-usable components may then be graded by their complexity. Simple components like multiplication and division (yes, once upon a time, these were programmed by programmers) are normally re-usable. Complex components used in advanced client server applications are not. Can we not sub-divide components until we get totally re-usable components? Not really. Many problems are not sub-divisible and others would, if sub-divided, increase the complexity of the system. And complexity would increase running time, a real problem in graphics systems, for example. Less of a problem in a single-user accounting system and no problem at all a word-processing system. If re-use is a function of

complexity and complexity is a function of running time, is it possible to derive rules that will guide us in the use or re-use of standard components?

We need a methodology that will sub-divide systems into efficient blocks or methods or components.

The Real-time Control System does just this. It accomplishes this by formalising the structure of an entire system or organisation in such a way that data flows are easily visualised and temporally analysed.

What is the Real-time Control System Model?

It is a reference model for intelligent real-time systems - loosely, systems in which the database is updated as events occur. It sub-divides organisation management into seven basic elements or layers: planning, execution, control, world modelling, sensory processing, user interface and communications - Albus [2]

The Real-time Control System clusters these elements into computational nodes that control specific subsystems, and arranges these nodes in hierarchical levels so that each level has a characteristic granularity. The levels are graduated in levels depending on the granularity of the level. The granularity of each level in the control hierarchy is derived from the characteristic timing and functionality of that level. For example, in a manufacturing environment, a dozen hierarchical levels are more or less standard. The actual timings for each level and the levels used may vary from business to business but the general principle remains unchanged. By sub-dividing the organisation = system in this way, we are able to determine the complexity of each sub-system. The Real-time Control System gives the system architect the ability to measure complexity along any axis he or she chooses. Returns on Re-use Investment are quick because system management can focus on specific problem areas. Yes, Re-use is an investment, in the same way that developing a new process or installing Statistical Quality Control in the factory is an investment.

The layers into which the work is divided are as follows:

Planning

This layer is where the strategic and tactical planning takes place. The time horizon at this level varies from one day to three or more years depending on the business.

Execution

This layer is where the commands are issued to the operating level based on planned requirements and task decomposition.

Control

This layer controls the execution of the process and is where value judgement is performed. The value judgements may be made by humans or by computers using Expert Systems.

Communications

This is where messages are passed between levels.

This layer includes electronic mail within the organisation and communication with customers and external databases outside the organisation. Local and Wide area networks will be used between all elements of the organisation so that data is continually kept current.

User interface

This layer includes the software that enables a manager to scan the business. The manager may be provided with standard reports and may be able to select additional information. The information will be presented in graphical (in its broader sense) format for easy assimilation. The layer will also offer analysis functions. The current situation will be presented in real-time as will future projections based on current events. In SynQontrol, a Statistical Process Control system, the latest release displays test results in graphical format, as they enter the system, in a Real-time Status Management Block that is continually available to management. From which it can determine the quality of the current and past production. These and other information displays enable management to control the company within its envelope. Managing proactively rather than reactively and conserving energy.

Sensory processing

This is the system picks up input. Depending on the layer different inputs are collected and processed. Some messages are passed to higher layers, while some are processed at the level at which they are received. It is worth noting, at this point, that 85% of all errors which occur in a system, computer or manual are system errors and cannot be fixed by the users. It is thus necessary to have direct and quick communication to the Command and Control layers. And good error recording routines so that you can analyse the problems when enhancement or extension time comes around.

World Model

This is the repository of static and dynamic data and the business rules of the organisation.

Static database

In this database are kept all or most of the fixed information of an organisation. Some data entities that will be kept: customers' addresses, recipes, routings, part data including CAD data and historical sensor data for analysis. This data will probably be kept in an Object Relational Database. Data enabling Virtual Reality functionality will also be kept in this database.

Dynamic database

In this database, which can easily be a conventional Relational Database, is stored the transient data that is the lifeblood of the business. This is composed of such data as orders, stock holdings, money, in other words those entities that change dynamically. It also includes external information like market research, benchmarking comparatives, political and economic analysis.

Business Rule set

The rules that govern the business are kept in this database. Note this particular database may, in practice, be a virtual database in that all the rules may be spread over different datasets. For example, the Business Rules relating to credit management may be kept in a separate dataset from those which relate to product quality. The format of the database dictates storage of Business Rules. In an Object Database the Rules are encapsulated with the data while in a Relational Model Rules may be stored as data in tables accessed by functions.

How are the levels organised?

I will now describe the 12 levels into which a system may be divided. 1 level is described in some detail, the others in less.

Level 12 - Strategic planning

This level schedules and controls the company for a period of about 3 years. Major capital equipment and new product decisions are made at this level. How does a 3 year planning horizon fit in with a real-time control system? The answer is simple. Decisions made today will affect the company's energy in the future. Energy is the sum of the men, machines, money, methods and measurements that is affected by the milieu in which the company operates. (Milieu = Environment. All of the others begin with the letter M.) If a company squanders its energy, it may be unable to compete in its environment in future. The layers found in this level are planning, execution, control, communications, user interface, sensory processing, and the World Model. Let us examine each of these in turn.

Planning has already been mentioned, the Execution layer implements what has been planned.. The Control layer monitors the situation and either takes corrective action or passes information to the Execution layer for display in the User Interface layer. The Sensors at this level are concerned with long-term planning and will include monitoring the political, economic and competitive environment . The Strategic Planning level is primarily concerned with Modification of the World Model especially the Business Rules.

Level 11 - Medium term planning

This level schedules and controls the company for a planning period of about 1 year. Some capital equipment decisions and product extensions are made here.

Level 10 - Short term planning

This level schedules and controls the company for a period of about 3 months Sales promotion decisions are made at this level.

Level 9 - Scheduling

This level schedules and commands the company for a planning period of about 1 month. Forecast orders are assembled into batches, raw materials are ordered and preventative maintenance planned. Sensor data at this level may be obtained from Market Research.

Level 8 - Factory

This level schedules and controls the factory for a planning period of about 1 week. Schedules are drawn up to meet delivery requirements for accepted and expected orders. Raw material deliveries are scheduled and instructions issued to suppliers to deliver at planned times and locations. Sensory Processing at this level includes bar-code scanners that provide data for raw material and warehouse management.

Level 7 - Shop

The level schedules and controls the activities of one or more manufacturing cells for about 24 hours. Commands are issued to cells to prepare schedules for the batches. The World Model at this level accesses the dynamic database of orders and the resources needed to process the orders.

Level 6 - Cell

This level schedules and controls the activities of several workstations for approximately 1 hour. Batches of ingredients are delivered to workstations and commands are issued to workstations to mix production batches. The World Model at this level accesses the database of equipment and materials needed for the immediate 1 hour planning period.

Level 5 - Workstation

This level schedules tasks and controls the activities within each workstation for about 5 minutes. The World Model at this level accesses the databases which contain works instructions, parts and tools held at the workstation and equipment readiness.

Level 4 - Equipment task

The equipment task level schedules task and controls the activities of a machine for a 30 second planning period. This level decomposes each equipment task into elemental moves for the subsystems that make up each resource. The World Model references the static database which for this level contains part attributes.

Level 3 - Elemental move

The elemental move level schedules and controls simple machine operations that require a few seconds. The World Model at this level will access the static database that contains part detail including some which may be in a CAD format.

Level 2 - Primitive

This level plans trajectories for tools, manipulators and inspection probes to minimise time and to optimise performance. The World Model at this level accesses the Static Database that supplies data for tool movement as related to items to be manufactured.

Level 1 - Servo

This level transforms commands from tool path to joint actuator co-ordinates. Planners interpolate between primitive trajectory points with a 30 millisecond look ahead. The World Model at this level contains data relating to state variables such as velocities, temperature and equipment reaction times. Data from reaction time sensors is plotted on a graph so that the technical manager becomes intuitively aware of potential problems such as hydraulic failure and can taken preventative action. Expert systems may highlight problems.

Standard component use related to levels.

The temporal granularity of each level will determine the possibility of use of standard components. At the lowest level where execution speed is important, for instance, in an aircraft control system, the developer would be working toward a custom solution that is tuned for efficiency. Components will probably be written in C++ or even Assembler. At the highest level where speed is not of great importance we would work towards a more generalised approach capable of solving many problems. In practice, we find that this is the case. Systems or sub-systems that are not time-critical tend to be the ones that have the greatest availability. Spreadsheets and graphs are good examples of off-the-shelf components. One would not think of flying a passenger aircraft using a spreadsheet package but making a purchase decision with one is normal.

The Real-time Control System, therefore, helps the system designer to plan re-usability. By modelling the entire organisation, the designer can determine the areas in which re-use will have the quickest payback.

How the Real-time Control System was used at Apron Services

The ground handling management system in use at Apron Services captures all customer aircraft movements and services provided to those customers. These services are cargo and baggage loading and unloading, passenger transport for those aircraft not parked at airbridges, assistance for sick passengers and aircraft towing. In addition it captures equipment, labour and fuel usage. Customer invoices as well as operational statistics are presented by the system. Monitoring of operations is currently at level 7. Extensions will include scheduling of operations (levels 6 and 7) and forward planning - level 8 and above. Payback for the system was less than 3 months.

Productivity in the organisation increased. Error rates in one particular area dropped from an average of 25% to less than 1%. Invoices that took up to 14 days after month-end to prepare are down to 1 day. The MIS department now has the ability to evaluate the productivity of operational departments and provides daily feedback to these departments in respect of errors and resource utilisation.

The system also provides South African Airways (a customer of Apron Services) with accounting and operational management information not available from their own information system.

A Relational database is used for the system. Some data would be more efficiently stored in an Object database but implementation of this is not a priority.

The implementation of the Real-time Control System followed the following phases:

1) User Requirements definition using the Real-time Control Model.
   Since aircraft handling is, at this point, largely a manual operation, the system was designed from level 5 upwards. Initial implementation was at level 7. This was chosen because it was the interface between the manual and the computer-based components of the system. Insufficient reliable data was available for the levels below 5. However, the system was designed to record data for future level 4 development. Once South African airports are mechanised to the extent of that of, say, Denver (was that a good example?) development will progress to lower levels.

   PLANNING at level 7 relates to staff and equipment availability and scheduling to optimise resource use.

   EXECUTION is not used at this stage owing to the volatility of the industry. 1 late 747 or bad weather can destroy a schedule.

   CONTROL is performed at this level by supervisors

   COMMUNICATION is done through a file server. Hand-held computers are not considered to be useful because of the labour intensive nature of the work. Process data is, therefore, captured from input forms.

   The USER INTERFACE is through the Relational data base using its query facilities or exception reports.

   There are altogether 74 exception and warning reports. The warning reports cover "Fuzzy" problems. An example might be that an Airbus300 used fewer buses than might be expected. Investigation of this possible problem revealed serious deficiencies in the bus control system. The depth of problem reporting has provided ample opportunity for customising reports in response to ad-hoc enquiries. Turnaround for ad-hoc enquiries is generally minutes.

   SENSORY INPUT is done manually by operating staff. However, designing the system with RCS means that upgrading sensors to, for example, barcode scanners or vehicle tracking devices will necessitate only changing that interface - the rest of the system will remain unchanged.

   The WORLD MODEL STATIC DATABASE contains amongst others, aircraft data like the number of passengers, the number of steps required and type of cargo handling. Other static data relates to customers (airlines), especially the aircraft used. Every registration number of every aircraft handled in the last 18 months is kept in the database. This data is used to assist input accuracy.

   DYNAMIC DATA is operating data like arrival times, registration numbers of equipment used and quantity of baggage handled.

   The BUSINESS RULES SET contains operating standards like baggage delivery and other handling times. Non-conformance with Business Rules is advised to management and staff for both corrective action and recognition of good performance.

2) Failure Mode and Effect Analysis. (Developed by Ford Motor Company.) This is an effective tool for estimating the cost and effect of potential problems. It provides focus to the system designer, ensuring that important potential problem areas are covered in the design We used Byzantine Failure Analysis - Turek and Sasha [6] to supplement the FMEA since we suspected that one or more of the processors or the software itself might fail and cripple the system.
3) Executable system modelling using Tool Abstraction proposed by Garlan, Kaiser and Notkin .[7]
   The finished system specifications were presented to one of the top programming houses in the country who programmed the system in 3 weeks. It went live immediately and has not failed during 18 months of use.

   Additions have been made to the original design extend its facilities and to accommodate changes in the environment. These have not impacted on the system because The Real-time Control System model methodology ensures design robustness and opportunity for re-use.

Summary

The challenge for South Africa in the next century will be to utilise our resources efficiently. This will entail structural changes in our organisations. Management will have to become immersed in the system. The system will be the organisation. Management will spend most of its time monitoring company health. Expert systems will analyse data arriving through the sensor layers, comparing this data to standards derived from internal and external research. Any variation in, for example, Productivity will manifest itself at an early stage and corrective action can be taken immediately. This is not futuristic, military aircraft are currently being designed to have an integrated structural health-and-usage monitoring system. "The system is used to perform real-time airframe-fatigue calculations and to monitor significant structural events and flight performance parameters." [1] Virtual Reality will enable the MD to "walk through" even remote plants on a daily basis. Access to data will not be restricted to senior management, the information system will allow everyone to know what everyone else knows. This will eliminate knowledge hierarchies in which knowledge is jealously guarded from other workers, staff and bosses alike.

The Real-time Control System will enable response to market forces to be made in hours rather than days or weeks as is the case now.

References

[1] Bae introduces 'smart' monitoring to EF2000 - Flight International - 1/7 May 1996 - p23.
[2] RCS: A Reference Model Architecture for intelligent control - Albus - Computer May 1992 - p56.
[3] Software Development - November 1995 - Industry Watch - Christine Comaford - p27.
[4] Software Development - August 1995 - Editor's Notes - Larry O'Brien - p7.
[5] Software Development - July 1996 - Software Estimation , Why Software Fails - Capers Jones - p49.
[6] The many faces of consensus in distributed systems - Turek and Shasha - Computer June 1992 - p8.
[7] Using Tool Abstraction to compose systems - Garlan, Kaiser and Notkin - Computer June 1992 - p30.

NEURAL NETWORKS FOR PROCESS PARAMETER IDENTIFICATION AND ASSISTED CONTROLLER TUNING FOR CONTROL LOOPS

Meredith McLeod and Vladimir B. Bajić

Centre for Engineering Research, Technikon Natal,
P.O.Box 953, Durban 4000, Republic of South Africa
Tel.: (+27) 31-204-2560, Fax: (+27) 31-204-2560
e-mail: meredith@umfolozi.ntech.ac.za

Abstract

The paper introduces a method for identification and assisted controller tuning for industrial process control loops suitable for an 'in-chip' solution. The method is based on utilization of neural network technology. It employs pattern recognition and exploits the nonlinear function approximation feature of neural networks. The method has the advantage of simplicity and sufficient accuracy. It is developed for on-line usage in closed-loop operations.

Introduction

Most process control loops are not well tuned (see Brown 1996). The proper control loop performance depends directly on adequate controller tuning. For that reason many standardized tuning methods are developed and used in practice (see for example Ziegler-Nichols 1942, 1943, Cohen-Coon 1952, Chrones *et al.* 1952, Yuwana and Seborg 1982, Shinskey 1979, Astrom and Hagglund 1988, Hang and Sin 1991, etc). Characteristic of all these methods is that they first attempt to determine some features of the model of the process to be controlled, and then provide appropriate controller tuning. Some of these standardized methods require testing on the open-loop process which is not always suitable as the process can be for example unstable. Another problem with the open-loop testing is that some processes do not allow breaking of the feedback due to operational constraints. Another group of methods uses closed-loop testing. Some of them induce specific oscillations in the system, while others consider the process response to specific test signals. All these methods are based on the analysis of process reaction from which the necessary information about the process is extracted and used in the next step for suitable controller setting.

ANNs have been used in numerous ways for control purposes (see Pham and Liu 1995, Warwick *et al.* 1992). Recently, ANNs have been utilized for different aspects of loop tuning. Several authors proposed their usage as auto-tuners of controllers (see Mamat *et al.* 1995, Ruano *et al.* 1992, Swiniarski 1990, Willis and Montague 1993, McLeod and Bajić 1996). For example, Mamat *et al.* (1995) have used ANNs to assist in the selection of optimal tuning of the controller after the process model has been determined on the basis of a modification of Yuwana and Seborg's (1982) method.

In this paper we will exploit the capability of static ANNs to encode the relationships between 168-element vectors representing 168 samples of the process output waveform and three crucial process model parameters affecting the shape of this waveform. The ANNs for this purpose will use pattern recognition. Once trained, an ANN provides rapid identification based on a single test waveform. When the process model parameters are obtained the optimal tuning of the PID controller is provided from a set of possible choices by ANNs. One approach where pattern recognition is used in connection with ANNs in the process of controller tuning is given for example in Valdebenito *et al.* (1995). The method proposed in this paper seems to be simpler, computationally more effective and more easily applicable to industrial applications. Also, the method proposed is suitable as the complete solution for EPROM implementation, where the trained ANNs will perform the task of process parameter identification and optimal controller tuning.

The method proposed is illustrated by a simulation example.

Preliminaries

The crucial part of the loop tuning in the method proposed in the paper is determination of the parameters of the process. For this purpose consider a classical negative unity feedback system with the PID controller and the plant in series connection in the forward branch (Fig.1).

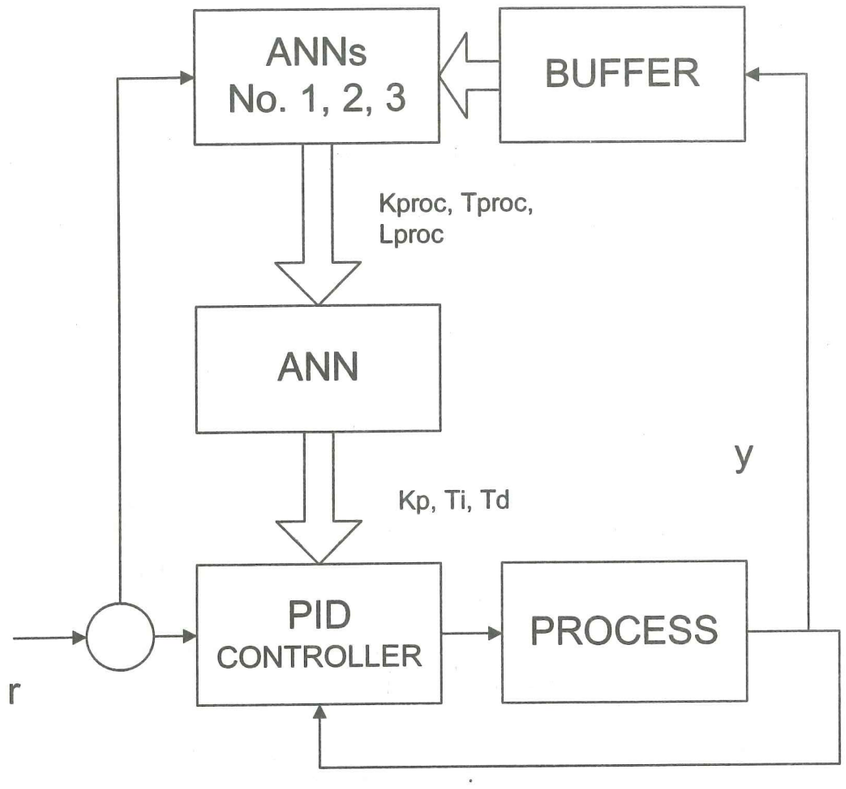


Fig. 1 Structure of the control system

Let r, u and y denote the reference input signal, the controller output signal and the measured plant output signal, respectively. It is assumed that the process model is governed or that it can be reasonably well approximated by the transfer function

$$G_{proc}(s) = \frac{K_{proc}}{T_{proc}s + 1} e^{-L_{proc}s} \tag{1}$$

The motivation for using this type of process transfer function is that many methods for controller tuning are based on such an assumption for the process model. The reason is that the step responses of many industrial processes roughly correspond to the response obtained by (1) as they are monotonic and self-saturating. Although there are many methods that can provide parameters of (1) in the closed-loop testing like the one given in Bologna *et al.* (1995), we will use ANNs to obtain this information. The main reason for this is that once trained, ANNs will provide convenient, computationally fast and sufficiently accurate answer about the process parameters and, based on this, optimal controller tuning. In principle, the whole 'solution' proposed in the paper can be memorized in EPROM and used like an additional one-chip unit attached to the already implemented PID controller. Our assumption is that initially the nominal value K^n_{proc} of K_{proc} is available. This value is usually easy to estimate, but the method proposed allows that it be arbitrarily chosen as a positive gain. The controller used for training of ANNs is a PID one governed by the operator equation

$$u = K_p e + \frac{K_p}{T_i s} e - K_p T_d \frac{N}{T_d s + N} y \tag{2}$$

where $e = r - y$ is the error signal. Controller parameters are K_p - the proportional gain, T_i - the integral time, T_d - the derivative time and N is the filtering constant.

Determination of parameters of (1) via ANNs is performed utilizing an on-line test with the control system in the closed loop. The flow-chart given in Fig.2 indicates the operation of the algorithm. There are three parameters of (1) to be determined, K_{proc}, T_{proc} and L_{proc}. They are determined in two successive phases that will be described later on.

Outline of the method for process parameter determination via ANNs

In applying the proposed method a known input pulse (Fig.3) is used as a calibrated 'stimulus' to produce an output waveform from the process under test in the closed loop with the PID controller (2) set to specifically determined parameters. The recorded response is then applied to one of the three ANNs which have been previously trained to associate the recorded waveform with the appropriate parameter value. This produces the value of a parameter from which the actual value of K_{proc} is determined. Then the PID controller is specifically retuned, and the same test is repeated, but the recorded waveform is presented to the other two ANNs to provide estimation of the parameters T_{proc} and L_{proc}/T_{proc}. The on-line experiment follows the next steps.

Step 1: Phase of K_{proc} determination. For test to start the closed-loop system has to be in the steady-state. The PID controller parameters are then set as given in the following table

K_p	T_i	T_d	N
$0.14798/K^n_{proc}$	10	1.332	10

The test signal shown in Fig. 3 is then presented to the system input (it is added to the signal r). The process response is recorded and presented to the ANN No.1, which produces the estimated value k equal to $0.4743 \cdot K_{proc}/K^n_{proc}$. From this one gets the actual value of $K_{proc} = kK^n_{proc}/0.4743$.

Step 2: Phase of determination of T_{proc} and L_{proc}/T_{proc} . At this stage the PID controller is retuned according to the following table

K_p	T_i	T_d	N
$0.312/K_{proc}$	10	1.332	10

This is a normalization of the process gain used to suit the next two ANNs which were trained with a training set generated with $K_{proc} = 1$. Then the same test signal as in Step 1 is applied to the system input in the same way as in Step 1. The process response is recorded and presented to the ANN No.2 which produces the estimated value T_{proc} and to the ANN No.3 which produces the value of L_{proc}/T_{proc}. In this way all three parameters of the process model (1) are determined.

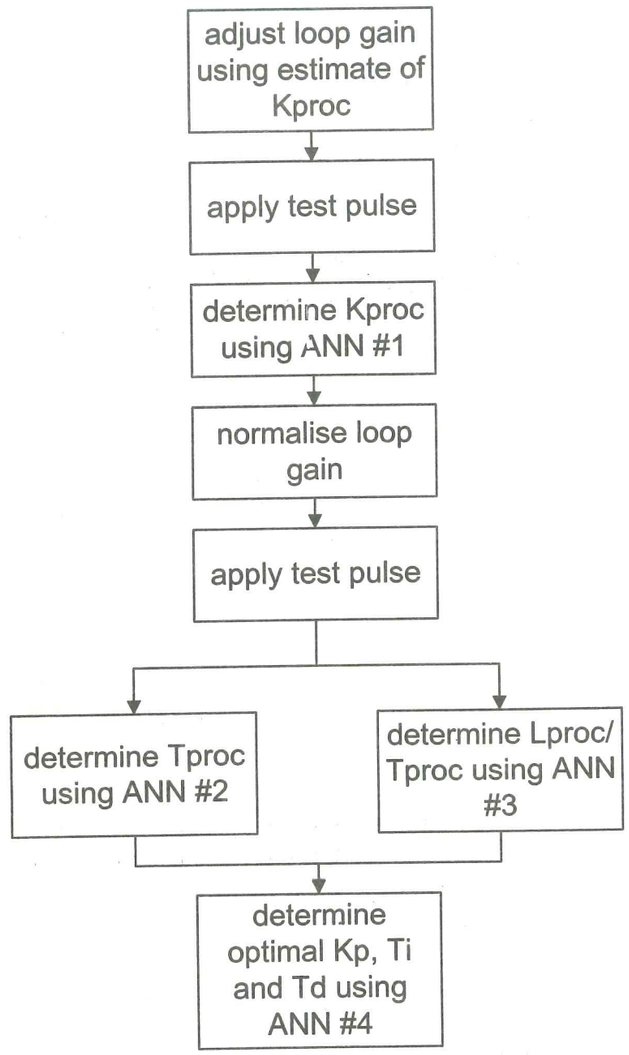


Fig. 2 Flowchart of the method

Preparation of the output signal waveforms for parameter identification

To prepare these waveforms to be used in the training set, the test signal shown in Fig.3 was used in conjunction with a PID controller (2) with the parameters set accordingly, and the parameters of the process model (1) varied in the appropriate ranges. The waveform chosen as the test input signal was selected as likely to generate a response illustrating the effects of varying the three process model parameters. The following setting of a PID controller is used:

$$K_p = 0.312,\ T_i = 10,\ T_d = 1.332,\ N = 10.$$

These values were selected to produce waveforms that were distinctly different but without excessive oscillation as the process parameters were varied. The range of values for parameters of the process (1) used for generation of the testing sets were the following:

$$K_{proc} \in [0.15,\ 1.5],\ T_{proc} \in [1,\ 7.8],\ L_{proc}/T_{proc} \in [0.3,\ 1.5],$$

i.e. the corresponding range for L_{proc} is [0.3, 11.7]. It was decided to use 168 samples of the output signal with the sampling interval of 0.2[s] for the length of 33.6 [s] for each input training vector. Examples of the typical output signal waveforms obtained are shown in Figs. 4, 5 and 6.

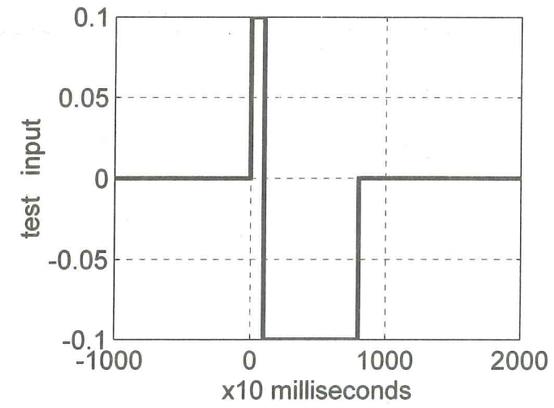


Fig. 3 Test signal

Neural networks for parameter identification - structure and training

Choice of ANN type. The choice of the ANN type was limited by the available software and hardware. Although initially we attempted to use ANNs based on the Levenberg-Marquardt Back-propagation Algorithm (BPLM), experiments showed that BPLM was not viable for this application due to large memory requirements. As we had MATLAB Ver.4.2c and Matlab's Neural Network Toolbox Ver.2.0c running on a PC with 64 MB RAM, the choice of radial basis function networks (RBFNs) seemed natural.

It is, in principle, possible to use one MIMO ANN to produce all three parameters of (1) at once. This approach was found to be beyond the capability of the PC and software being used for these experiments due to memory limitations in the training phase of MIMO RBFN. As a consequence the separate ANNs for determination of each of the three process parameters were used. It should be noted that the memory constraint experienced with MIMO RBFNs existed only in the training phase. Once trained, the ANN can be invoked with a smaller memory requirement. Since the final goal is to utilize them as trained networks, this approach that employed more than one ANN was considered acceptable.

Training sets. Initially an attempt was made to use a training set of 1000 vectors obtained from process response, taking only 10 values of each of the three model parameters ($10 \times 10 \times 10 = 1000$). This training set was found to work only for K_{proc}. It was necessary to use a separate training set composed of 900 vectors obtained using 30 values from the range for T_{proc} and L_{proc}/T_{proc}

($30 \times 30 = 900$). The problem was therefore taken up in two stages. Firstly, a single radial basis function network (ANN No.1) was trained to output the value of K_{proc} in the range [0.15, 1.5] when presented with a waveform produced by the process model (1) with unknown parameters and controlled by a 'reference' PID controller. Secondly, ANN No.2 and ANN No.3 were trained to output T_{proc} and L_{proc}/T_{proc} respectively, once the process gain K_{proc} had been normalized to unity. The test signal of Fig.3 was applied 1105 [s] after a unit step input signal r, when the steady state of the process output y had virtually been established. Then the test signal was applied. The response waveforms were recorded as 168 data points over 33.6 [s] from the time of application of the test signal. The 'reference' controller parameters were selected so as to produce stable but unique waveforms for all combinations of process parameters.

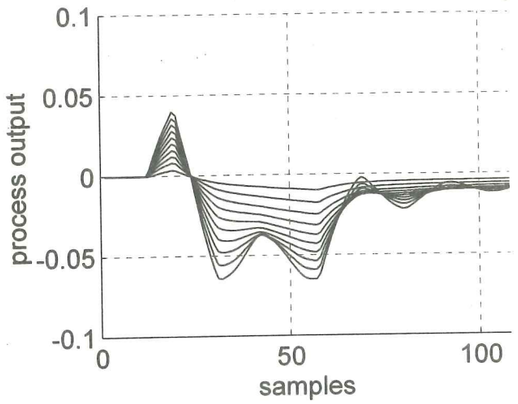


Fig. 4 Response for different values of K_{proc}

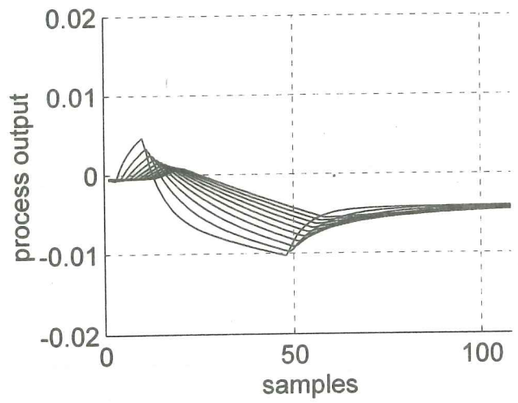


Fig. 5 Response for different values of T_{proc}

Training results of ANN No.1. It was found that the clearly defined amplitude changes as K_{proc} was varied permitted effective training using as few as 10 values of K_{proc} in the range under consideration. Ten values each of T_{proc} and L_{proc}/T_{proc} over the ranges under consideration were combined with the 10 values of K_{proc} to produce a 1000-vector output training matrix. The input training matrix was produced using these values in the closed-loop simulation with the 'reference'

PID controller and recording the process output waveforms. This produced 1000 vectors with 168 elements. The RBFN training took 57 [min] to reach a sum-squared error (sse) of 0.0003. There were 121 hidden layer neurons (HLNs). Subsequent testing using 100 randomly chosen triplets (K_{proc}, T_{proc}, L_{proc}) produced a maximum absolute error (mae) of 0.938% and a standard deviation (std) of 0.249.

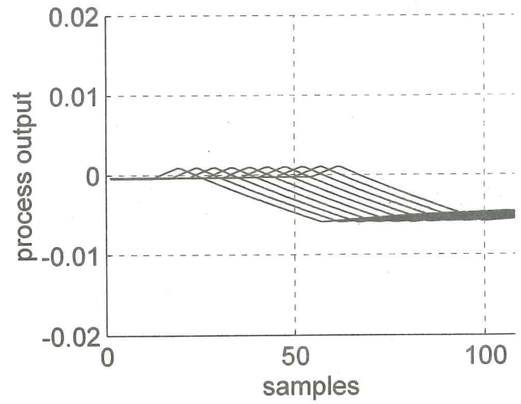


Fig. 6 Response for different values of L_{proc}/T_{proc}

Training results of ANN No.2 and ANN No.3. Because difficulties were experienced in training RBFNs to recognize T_{proc} or L_{proc}/T_{proc} from the waveforms for the $10 \times 10 \times 10$ combinations used in ANN No.1 it was decided to keep K_{proc} constant as $K_{proc} = 1$ and use 30×30 combinations of T_{proc} and L_{proc}/T_{proc}. The results can be summarized as follows:

ANN No.2 (determination of T_{proc}): It required 37 hidden layer neurons, 10 [min], $sse = 0.3$. Test with 100 randomly selected triplets (K_{proc}, T_{proc}, L_{proc}) resulted in $mae = 3.30\%$, $std = 0.676$.

ANN No.3 (determination of L_{proc}/T_{proc}): 246 hidden layer neurons are required in the network, 185 [min], $sse = 0.0003$. Test with 100 randomly selected triplets (K_{proc}, T_{proc}, L_{proc}) produced $mae = 1.13\%$, $std = 0.309$.

All RBFNs used gaussian activation functions in the hidden layer with a linear function in the output layer.

Method for optimal controller tuning via ANN

There are several methods for tuning of a PID controller when the model (1) of the process is known. Some of these are mentioned in the Introduction. However, the ANNs give an opportunity to memorize the optimal tuning for the controller for different optimization criteria and different constraints under the wide range of process parameter values. One such ANN approach is given in Mamat *et al.* (1995) where only one optimization criterion is used for derivation of controller parameters. What we propose is an extension of that approach, where a set of optimum controller settings is generated for the range of parameters of the process model (1) and for a number of different tuning requests commonly used in practice, e.g. optimal disturbance rejection, optimal set-point tracking, combined optimal disturbance rejection and set-point tracking, etc. Then, one or more ANNs are trained to match the controller parameter values to the process parameter values. From the practical viewpoint this should not be a problem. Then the ANN can choose, according to the specific need, the correct controller tuning. The specific need for a particular loop can be selected by the operator from the choices provided. This part of the application does not go beyond the standard application of static ANNs and will not be discussed further.

Example

This example demonstrates the procedure of process model identification and controller setting using ANNs. The quality of disturbance rejection of the neural network tuned PID controller is then compared with the quality of disturbance rejection of a system tuned according the Cohen-Coon method.

For a practical assessment of the method it was necessary to choose an example of an unknown plant. A simulation was set up using a fourth order process described by

$$G_p(s) = \frac{0.5}{(s+1)^4} e^{-s} \tag{3}$$

The nominal value of the process gain is guessed to be $K^n_{proc} = 1$. The 'reference' PID controller was set as outlined in Step 1, the test signal of Fig.3 was applied to the system input and the output of the process was recorded. The recorded waveform was then applied to ANN No.1 yielding $K_{proc} = 0.5006$. Then the PID controller was retuned according to the requirements outlined in the Step 2 and the test signal was applied again. The resulting waveform was applied to ANN No.2 and ANN No.3 to yield $T_{proc} = 2.18$ and $L_{proc}/T_{proc} = 0.974$.

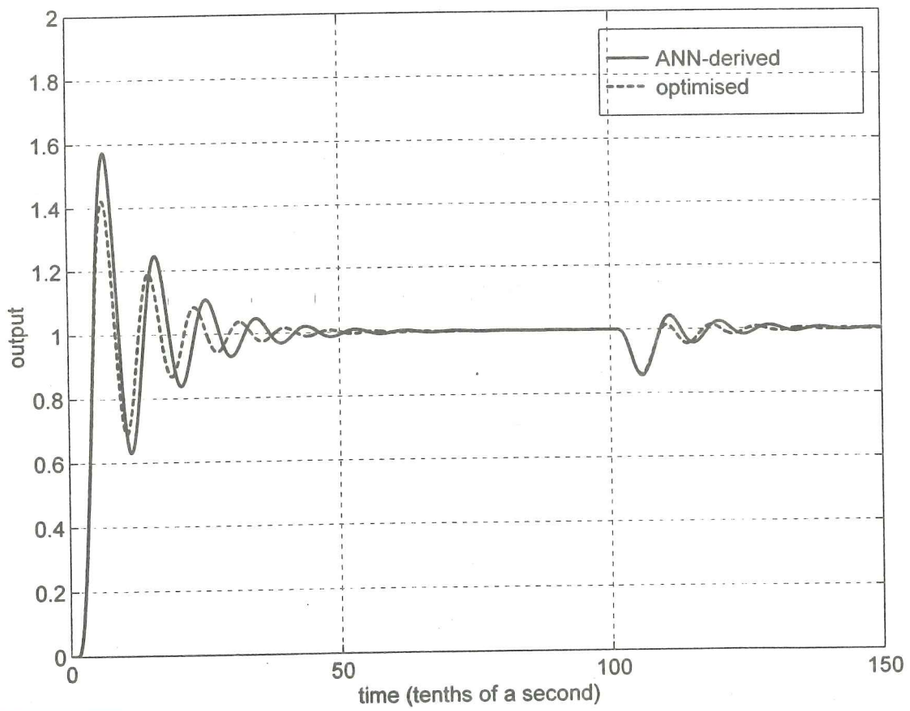


Fig. 7 Closed-loop step and disturbance response

The tuning of the controller that provides good rejection of step-type disturbances is then determined via ANN as given in the next table

K_p	T_i	T_d	N
1.6578	4.0026	0.6576	10

A simulation was then set up using these PID parameters and the exact fourth order transfer function (3) of the plant, as well as using the PID parameters determined by Cohen-Coon method. The results are shown in Fig.7. The disturbance had the step size of -0.5 units. As can be expected the set-point tracking is not good in either case, but we point out that the controller was not tuned for good set-point tracking. However, the disturbance rejection of the actual fourth-order plant (4) in the closed loop with the PID controllers tuned via ANN and by Cohen-Coon method is very good. The dashed line shows the response when controller settings is based on the Cohen-Coon method for the equivalent model (1) of the actual fourth order model (4) of the process. The solid line shows the response when the ANN-derived controller setting is used. It can be seen that the ANN-derived settings of the controller produce disturbance rejection very close to that obtained with the Cohen-Coon method. This confirms the validity of our approach for process parameter identification and assisted controller tuning.

Conclusions

The paper proposed a new ANN based method for identification of a process model and assisted controller tuning. The method gives directly the values of the process model parameters. These parameters are then utilized for optimal controller tuning. The method possesses simplicity and applicability for a wide range of process parameter values. It is suitable as a complete 'in-chip' solution that can be loaded to EPROM and utilized as a separate unit attached to the implemented controllers. At this stage the solution can be improved by utilizing only one ANN instead of three currently used for process parameter determination, but this seems to be only a technicality. A potential domain of application of the method is in process control field applications, as the ANNs involved can be trained in advance.

References

K.J.Astrom & T.Hagglund (1988), "*Automatic tuning of PID controllers*", Instrument Society of America (ISA).

E.Bologna, C.Tait & V.B.Bajić (1995), "Closed-loop process identification for auto tuning of PID controllers", *Proceedings of the Intelligent Control Symposium*, University of Natal, 20 June, Durban, RSA.

M.Brown (1996), "The state of control in South Africa", *Proceedings of the 2nd Application Symposium of the SAIMC*, July 4, Durban, RSA.

K.L.Chien, J.A.Hrones & J.B.Reswick (1952), "On the Automatic Control of Generalized Passive Systems", *Transactions of ASME*, Vol.74, pp.175-185.

G.H.Cohen & G.A.Coon (1953), "Theoretical considerations of retarded control", *Transactions of ASME*, Vol.75, pp.827-834.

C.C.Hang & K.K.Sin (1991), "On-Line Auto Tuning of PID Controllers Based on the Cross-Correlation Technique", *IEEE Transactions on Industrial Electronics*, Vol.38, No.6, pp.428-437.

R.Mamat, P.J.Fleming & A.E.B.Ruano (1995), Neural networks assisted PID autotuning, *Proceedings of the Second AIAI Conference on Industrial Automation*, Vol.II, pp.849-854.

M.McLeod & V.B.Bajić (1996), "Neural network assisted tuner for tracking improvement in process control", The AMSE International Conference, Brno, Czech Republic, September 10-12.

D.T.Pham & X.Liu (1995), *Neural Networks for Identification, Prediction and Control*, Springer-Verlag, London.