



**DNMI**

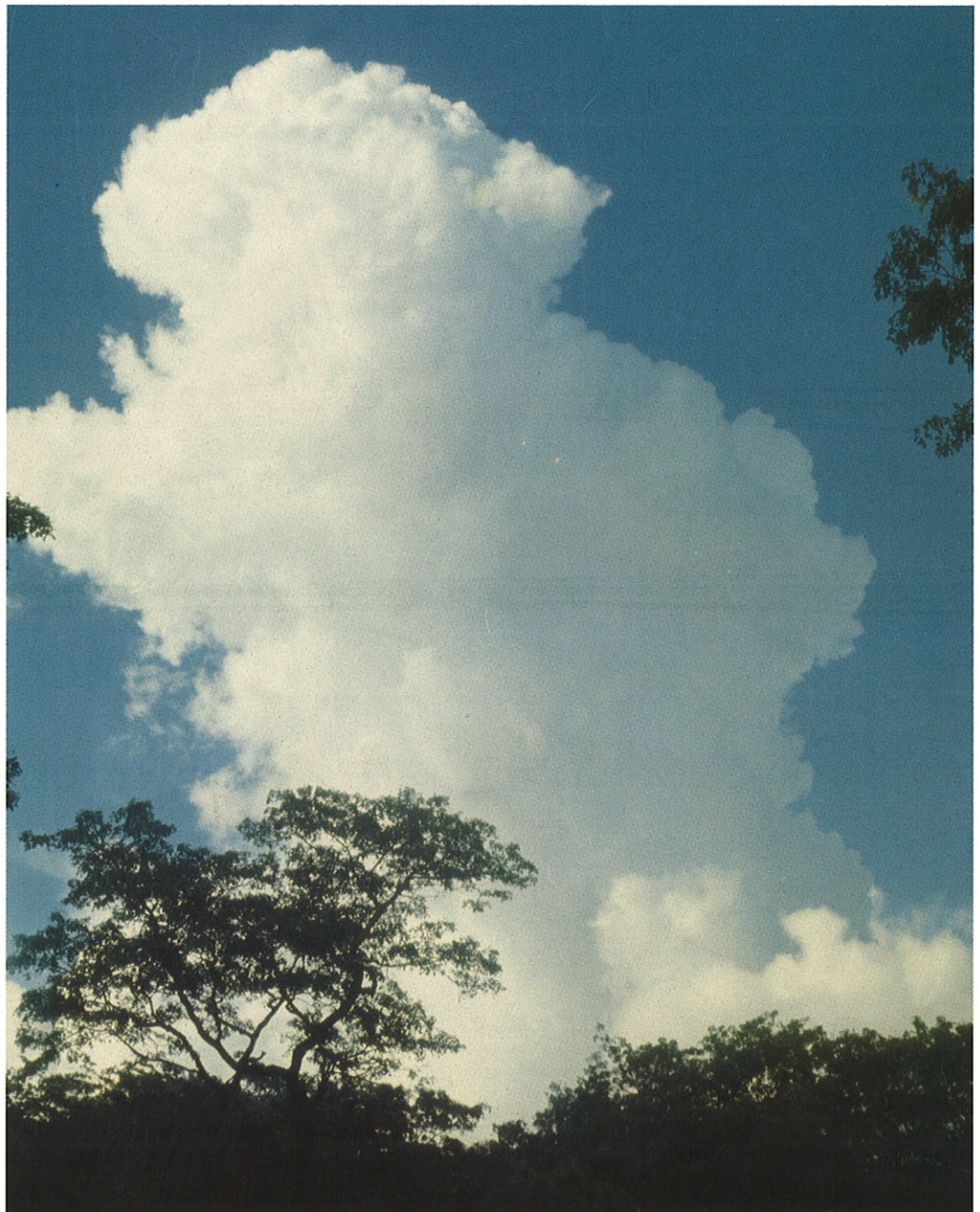
Det norske meteorologiske institutt

# **KLIBAS research notes volume 1**

**Report no. 01/99**

**Petter Øgland**

**KLIMA**



# DNMI - REPORT

ISSN 0805-9918

NORWEGIAN METEOROLOGICAL INSTITUTE  
BOX 43 BLINDERN N-0313 OSLO

PHONE: +47 22 96 30 00

REPORT NO.

01/99 KLIMA

DATE

Jan 4 1999

## TITLE

KLIBAS RESEARCH NOTES VOLUME 1

## AUTHOR

Petter Øgland

## PROJECT CONTRACTOR

DNMI - Climatology Division

## SUMMARY

In this volume eight research notes dating from November and December 1998 are presented. All notes are related to on-going research and development of the KLIBAS climatological database system.

The first note concerns the problem reducing internal time friction for the SYNOP data collection. The second note issues the problem of reading PIO («PC In the Observation Services») observations into KLIBAS. The third note is concerned with the dataflow of the KLIBAS system, addressing problems related with the updating of Semi-Automatic Weather Stations (SAWS) in the main data processing routine (ALV) by use AWS observations from the AWS data processing routine (ALA). The fourth note continues along this path by facing problems concerning the robustness of an AWS data transport program. The fifth note concerns problems related to another AWS data transport program. In the sixth note, a user interface problem for the AWS statistics is discussed.

The seventh note addresses a problem having to do with the design of a quality control program (CONTSYN1) in the main data processing routine, while the eighth note comments on a problem having to do with data transport for SAWS in relation to the PIO data processing routine.

## KEYWORDS

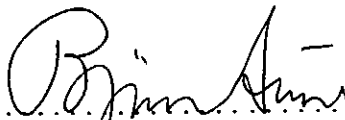
1. Climatological databases
2. Meteorological data collection
3. Meteorological quality control
4. SYNOP, PIO, AWS, SAWS

## SIGNATURE



Petter Øgland

Research Scientist



Bjørn Aune

Head of Climatology Division

## Table of Contents

Foreword .....	3
Reducing the internal time friction for the SYNOP data collection at DNMI .....	6
Reading PIO observations into the KLIBAS database system at DNMI .....	11
Dataflow in the KLIBAS database system at DNMI: Updating SAWS in ALV with AWS observations from ALA .....	14
Facing a problem of robustness on the MND2HLA AWS data transportation program in the KLIBAS database system at DNMI .....	16
Comparing collapse history for the MND2ALA and MND2HLA data transportation programs in the KLIBAS database system at DNMI .....	18
Problems in AWS statistical reports due to running Oracle i different environments for the KLIBAS database system at DNMI .....	21
Deciding the future of the CONTSYN1 data check for the KLIMA control routine at the Climatology Division at DNMI .....	22
Reading and converting SAWS weather observations V1/V2/V3 into the KLIBAS database system at DNMI .....	24

## Foreword

Late 1989 it was announced that a new climatological database system, later coined KLIBAS, should be planned and implemented in order to replace the previous solution (Moe, 1995). A group of representatives from all divisions was assembled, and requirements for the computer and database system was outlined.

It was decided that the implementation of the system should be done with limited resources. At the Climatological Division a group of approximately eight persons ("the database group") was established (Moe et al, 1991), first to accomplish the requirements for climatological data, and later to design and implement the system.

Requirements were sent potential vendors late 1990. Evaluations and discussions of the offered systems took place during spring 1991, and an Oracle database performing on Silicon Graphics computers was decided upon summer 1991 (Moe, 1995).

The work of the database group was organised in projects, giving goals, estimations, work packages and schedules, especially during the design phases (Øgland et al, 1992; Moe et al, 1994). As implementation commenced, during the second half of 1993, unpredictable problems arose, and work was hence done in a more experimental manner (Moe et al, 1994; Øgland et al, 1994).

The first part of the KLIBAS database system to be implemented was the module responsible for 1.5 gigabytes of precipitation data. During the first half of 1994 routines for handling the SYNOP data were added, along with a prototype version of the routine for handling data from climatological weather stations (Øgland et al, 1994). During the second half of 1994 the database was updated from Oracle 6 to Oracle 7, the operative programs and systems were refined (Øgland et al, 1995).

In 1995 about 12 gigabytes of disk was configured for the database, and a large amount of historical weather observations, maritime observations and plumatic observations were inserted into the database (Øgland et al, 1996). The Oracle database core was updated from version 7.0 to 7.1. Much of the work of 1995 did also involve security routines, backup, the archive log of the Oracle database system, technical equipment such as installment of PC's, when to dispatch the old ND-788 system and geographical information systems (GIS).

By the summer of 1996 KLIBAS contained data and routines for handling precipitation stations, traditional weather stations and synop, Aanderaa automatic weather stations (AWS), plumatic precipitation intensity stations, maritime weather stations and metar aeroport weather stations (Moe et al, 1996). Different kinds of statistics computer programs were made as a part of KLIBAS. During the second half of 1996 all routines were refined, and a significant amount of historical observations from Aanderaa AWS were inserted into KLIBAS (Øgland et al, 1997).

In October 1997 the database project was officially terminated. During 1997 all relevant historical weather observations had been moved from outdated systems and inserted in KLIBAS or related systems. Most of the work done by the database group was now restricted to maintenance and refinement of routines. It was furthermore announced that a new database project would commence in 1997, moving KLIBAS to a new database server and updating the database structure in order to handle demands unconsidered at the time when the KLIBAS database project was initiated (Øgland, 1998).

In 1998 improvement and refinement of the KLIBAS routines have continued. The system now consists of more than 120 computer programs, and during the year system documentation of updates or established of new programs gave rise to 69 KLIBAS system documentation reports (Øgland, 1999). Many KLIBAS programs are now systematically being logged and a great amount of time is being put into keeping the system alive.

During the two last months of 1998 eight research notes addressing different aspects of problems arising in the KLIBAS system were written under the assumption that better solutions would be found for the system if the problems were analysed in a formal manner, making it possible to communicate problems that are being found in a more systematically than before and make sure that problems are sufficiently specified and understood before a solution is attempted.

"Reducing the internal time friction for the SYNOP data collection at DNMI" was the first problem given this systematic treatment. The note was written in its final form on November the 27th and the

SYNO\_INN program was consequently adjusted according to analysis and suggestions.

The second problem addressed was "Reading PIO observations into the KLIBAS database system at DNMI". This resulted in the detection of an error in the PIO database system of KLIBAS which was consequently corrected.

The next three notes are related to problems with the AWS computer programs. The note "Dataflow in the KLIBAS database system at DNMI: Updating SAWS in ALV with AWS observations from ALA" resulted in an update of the ALA2ALV program documented in KLIBAS-report no. 66/98. Results from the note "Facing a problem of robustness on the MND2HLA AWS data transportation program in the KLIBAS database system at DNMI" ended up as KLIBAS-report no. 67/98, while the problem discussed as "Comparing collapse history for the MND2ALA and MND2HLA data transportation programs in the KLIBAS database system at DNMI" resulted in KLIBAS-report no. 68/98.

After these three major updates in the KLIBAS system, a lesser but still important problem concerning the user interfaces was discussed as "Problems in AWS statistical reports due to running Oracle in different environments for the KLIBAS database system at DNMI".

The note "Deciding the future of the CONTSYN1 data check for the KLIMA control routine at the Climatology Division at DNMI" did on the other hand result in a total reprogramming of one of the central quality control programs. The result was internally published as KLIBAS-report no. 69/98.

The final note "Reading and converting SAWS weather observations V1/V2/V3 into the KLIBAS database system at DNMI" caused reprogramming for the PIO\_INN system, but in this case not drastic changes of the type that would imply a need for updating the system documentation.

These eight KLIBAS research notes make the body of this first volume of KLIBAS research notes. The prime reason for documenting the analysis of problems is to hopefully gain long term advantage of a better understanding of how and why KLIBAS fails. Even though only special cases of problems with the KLIBAS system is discussed, some of these cases may exemplify general problems that can arise in other systems and contexts.

Petter Øgland  
Blindern, January 4th 1999

#### References:

- Moe, M., Iden, K., Kjensli, P.O., Kristiansen, S., Lystad, S.L., Nordin, B., Vidal, Å.M. and Aasen, T., 1991. *Database/maskin prosjektet i Klimaavd. 1990-1991. Informasjonsmodell, flagging og kontroller. Status pr 30.06.91.* KLIMA-report no. 32/91, DNMI, Oslo.
- Moe, M., Øgland, P., Vidal, Å.M., Aasen, T. and Kjensli, P.O., 1994: *Databaseprojektet i Klimaavdelingen. Status pr 31.12.1993.* KLIBAS-report no. 03/94, DNMI, Oslo.
- Moe, M., 1995: *KLIBAS - The DNMI Climatological Database System.* KLIMA-report no. 22/95, DNMI, Oslo.
- Moe, M., Kjensli, P.O., Vidal, Å.M., Øgland, P., and Aasen, T., 1996: *KLIBAS - status 30.06.1996.* KLIBAS-report no. 13/96, DNMI, Oslo.
- Øgland, P., Iden, K.A., Kjensli, P.O., Lystad, S.L., Moe, M., Nordin, B., Vidal, Å.M. and Aasen, T., 1992: *Databaseprojektet i Klimaavdelingen. Status pr 23.12.1992.* KLIMA-report no. 53/92, DNMI, Oslo.
- Øgland, P., Kjensli, P.O., Moe, M., Vidal, Å.M., and Aasen, T., 1994: *Databaseprojektet i Klimaavdelingen. Status pr første halvår 1994.* KLIBAS-report no. 24/94, DNMI, Oslo.
- Øgland, P., Kjensli, P.O., Moe, M., Vidal, Å.M., and Aasen, T., 1995: *Databaseprojektet i Klimaavdelingen. Status pr årsskifte 1994/95.* KLIBAS-report no. 06/95, DNMI, Oslo.
- Øgland, P., Kjensli, P.O., Moe, M., Vidal, Å.M., and Aasen, T., 1996: *Databasegruppen 1995.* KLIBAS-report no. 01/96, DNMI, Oslo.

- Øgland, P., Kjensli, P.O., Moe, M., Vidal, Å.M., and Aasen, T., 1997: *Referater fra møter i databasegruppen 1996*. KLIBAS-report no. 03/97, DNMI, Oslo.
- Øgland, P., 1998: *Arbeid i databasegruppen 1997*. KLIBAS-report no. 07/98, DNMI, Oslo.
- Øgland, P., 1999: *KLIBAS process improvement December 1998*. KLIBAS-note no. 01/99, DNMI, Oslo.



## Reducing the internal time friction for the SYNOP data collection at DNMI

*Petter Øgland*

Norwegian Meteorological Institute  
November 27th, 1998

### ABSTRACT

Due to repeated changes in the SYNOP\_INN data collection system, including the adding of statistical control methods for making it possible for the system to do self diagnosis and prevent uncontrolled breakdown, each run of the system seem to take alarmingly longer time from month to month. Although the reason for this development is not fully understood, various aspects of it is analysed and explained, leading to suggestions for further implementations and research.

### SYNOPSIS data collection

The purpose of the SYNOP\_INN data collection system is to read binary GTS files that are updated every 5 minutes into an Oracle RDBS as often as necessary, at the moment every 10 minutes. The present SYNOP\_INN program has been developed through several versions.

Using data from realtime synop data flow for insert in the climatological data base was discussed in 1985. Andresen et al. [1] wrote at the time a report addressing quality control issues in relation to observations being digitalised outside the Climatology Division.

The dataflow then implemented consisted on only reading observations relevant for long time storage in the Climatological database system on the ND computers. The dataflow was summarized in further detail during the autumn of 1993 by Øgland, Aasen and Vidal [2] as a part of the plans for the implementation of a new data collection system to be build for an Oracle RDMS on a SGI platform.

The first version of the SYNOP\_INN program, reading all observations from zone one on the synoXX-files into Oracle database tables, was implemented and put in operation in February 1994 (Øgland [3]) and revised due to upgrading the Oracle database system from version 6.0 to 7.0 in November 1994 [4], and revised once again by the end of the year [5] with a final revision 2.0

in April 1995 [6].

Runtime statistics from the system were systematically collected and summarised under the name GTS2FIFO in monthly reports from June 1995 and onwards (Øgland [7]). The name GTS2FIFO was chosen due to a FIFO-construction modelled after the work by Schøyen [8] on administrative data.

In October 1996 the SYNOP\_INN program was completely rewritten [9], later versions up to the present [10-15] being grown out of this code.. PP In figure 1 runtime statistics in terms of median value of seconds for each month is plotted. The change of from version 2.0 to 3.0 in October 1996 is marked by a circle. The system appears faster after this change, but still there seems to be a tendency of runtime increasing by the month. An explanation for this growth in time, that is mentioned in the SYNOP\_INN reports, is that the datatables are growing and each insert consequently takes longer time.

From February 1996 and onwards the definition used for runtime logging was changed, now logging the time used for inserting each syno-file instead of logging a session or group of files. This means that runtime statistics are not comensurable before and after this. The new curve show the same alarming characteristics, however, as the old one in that it is rapidly increasing. In November 1998 this caused consecutive runs of the program to bump into each other, causing warning

and error messages with a prospect of total collapse within short time.

Standard deviation: 138.8 seconds

Number of runs: 13876

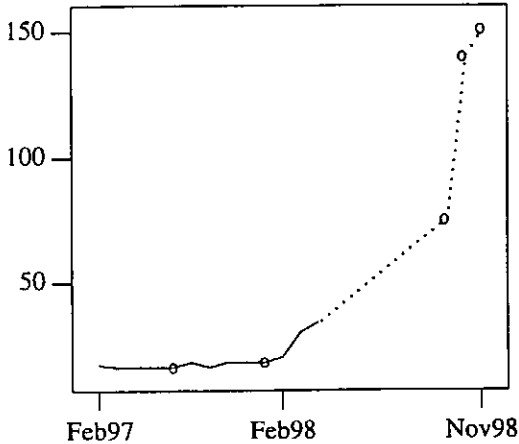


Fig 2. Average runtime in seconds

As with the GTS2FIFO versions of the SYNO\_INN program, figure 2 shows the increase in runtime averages from month to month. The revisions 3.2, 3.3, 3.4, 3.5 and 3.6 of SYNO\_INN are marked by circles on the curve. Unfortunately statistical values for the interval May 1998 to October 1998 have been lost due to failure while testing version 3.6 of the program in early November, but, nevertheless, remaining values seem give a rather non-blurred picture of the critical development of the program.

The plot in figure 3 shows the frequency distribution of  $T$  for November 1998. Characteristics of the distribution is a long right hand tail and three peaks.

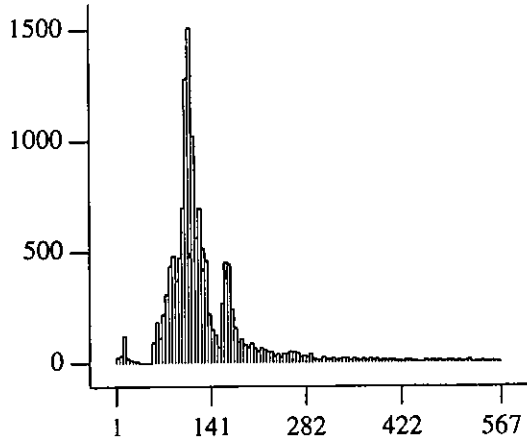


Fig 3. Runtime frequency distribution

The distinct peaks in the frequency distribution in figure 4 could be explained by a change of mean value for the process  $T(s)$  during the month. A time plot consisting of daily averages of  $T$  is presented in figure 4 does, in fact, seem to support this theory.

**Measuring runtime statistics**

In order to give objective measurements of how long each particular run takes, the execution time is calculated by the program as it runs and added to a log file.

The runtime  $T(s)=t_1(s)-t_0(s)$  of SYNO\_INN is defined in seconds without decimals where  $t_0(s)$  and  $t_1(s)$  correspond to the SGI internal clocktime read by the program started at  $s$  as soon as it starts to run and when it is about to terminate.

By the end of the month, in order not to overuse disk space, the measurements of the months are replaced by a runtime average and a standard deviation. Statistics for November 1998 amounts, so far, to the following:

Shortest time of execution:	1	second
Longest time of execution:	2970	seconds
Runtime average:	151.1	seconds

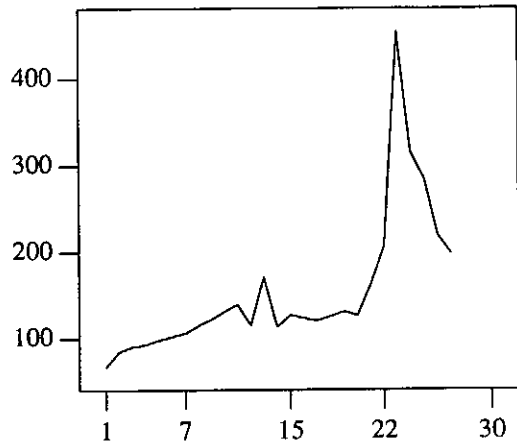


Fig. 4 Day by day runtime averages

The information in figure 4 shows that there is a significant time gradient at the time of when the version 3.6 of the system was put in operation. After this peak the program goes back to more normal behaviour although it still seems to be running slower and slower for each consecutive



day of the month.

### Possible sources for gross time consumption

The SYNO\_INN program makes use of several Oracle datatables and data files for temporary storage, logging and process control. There are three datatables that are being updated with each observations that is read by the program:

1. *TELE* contains only basic observations at 00, 06, 12 and 18 UTC. Observations are not systematically deleted from the table, so there is a significant accumulation of data on a daily and monthly basis.
2. *SYNOP* contains all observations found on the synoXX-files. As observations are not being systematically deleted on this table either, there is here an even greater accumulation of data making the table more complex and timeconsuming to update.
3. *SYNOP2* contains only the last 90 days of SYNOP with some additional observations from the automatic weather stations that are not a part of the synop network. As indices are updated the table may demand more time on each update as the months go on although the amount of data is kept constant.

As can be seen from figure 5, the SYNOP data table contains more than twice as much data as the SYNOP2 table, the TELE table somewhat less data than the SYNOP table but still significantly more than the SYNOP2 table. The SYNOP table contains data, from one or two up to 24 observations pr station pr day, from April 1998 until the present month. The TELE table contains data from July 1997 with a maximum of four observations each day, but the data set does not contain a complete list of stations until February 1998.

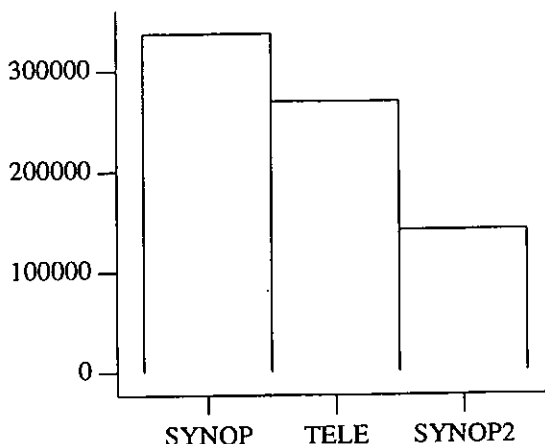


Fig 5. The number of rows presently in each table

Another possible source for time consumption may be the log files that are being updated on every run of the program. The files divide into two group, one group of files that is being used for system log and one group of files that is being used for data control. The group of system files consist of:

- a) *The ERR files.* These files are being updated whenever a system warning or system error is recognized by the SYNO\_INN program. More warnings generate larger files, and as the files are being read on each run, the total ellapse of time may increase.
- b) *The FIX files.* These are rather small files compared with the ERR files and contain mostly manually updated information on system changes whenever SYNO\_INN is being altered or reprogrammed.
- c) *The LOG file.* This is a file that grows with each run, and as the SYNO\_INN program is run typically around 10000 times in a month, this file would inflict on the daily growth in time consumption. While it presently contains log from all runs of SYNO\_INN, in the period Mars to September 1998 it only contained log from the synoXX files containing observations at 00, 06, 12 and 18 UTC.

While the size of the files a) to c) above are only depending on the performance of the SYNO\_INN program, the size of files d) to f) below is a function of the quality of the observations in the synopXX files which means that the size of the files are not as controllable as a) to c).

- d) *The TST files.* These files are used for logging data errors or data problems that have to be handled while running SYNO\_INN. In the TELE, SYNOP and SYNOP2 tables there are restrictions on the meteorological values to be stored, and values out of range will not be inserted. Whenever such a case is found, or the time of observation does not correspond with definitions in data table TELE\_PARA, the instance is logged on a TST file.
- e) *The TMP files.* In order to make statistical process control (SPC) there is a need for counting and remembering how many errors or problems that have been detected each day. The TMP files are updated and contain the total number of problems and the distribution of problems among stations for each run.
- f) *The TXT files.* The output from the SPC is presented in curves and statistical values for daily check and monthly publication in the Process Improvement reports. The size of each TXT file does, however, not change radically neither by day nor month.

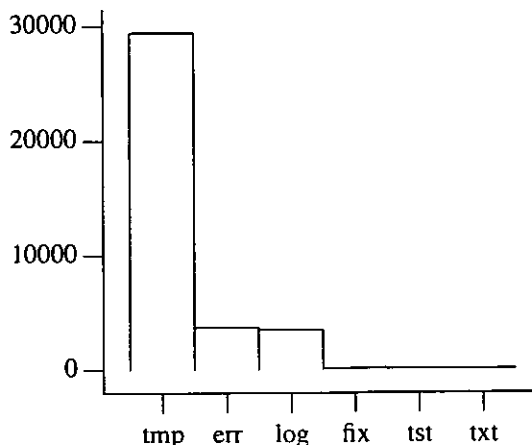


Fig 6. Size (blocks) for groups of files

As is visualized in figure 6, the size of the tmp files are almost 8 times as big as the err files. The log file is also quite large, almost as large as the sum of the twelve err files. On the other hand, the total size made up from the sum of the three last groups is less than 13 times as small as the log file, and should play an insignificant role in the time consumption of the program.

### Conclusions

In order to find out how each of the plausible causes for runtime fatigue may infect the program, daily measurements of file size and table size should be recorded in order to perform regression analysis.

From the present analysis, the most reasonable thing to do is to find a way to reduce the size of the tmp files. However, as the program history shows, there is a tendency of increase in time consumption, not having anything to do with the tmp files, so daily recording of file sizes and table sizes should be done anyway.

### References

- [1] Andresen, L. et al. *Rapport nr. 1 fra datakontroll-gruppen.* KLIMA work note no. 41, DNMI, Oslo, 1985.
- [2] Øgland, P., Aasen, T., Vidal, Å.M. *Data inn - spesifikasjonsrapport.* KLIBAS-report no. 01/94, DNMI, Oslo, 1994.
- [3] Øgland, P. *Innlasting av synoptiske data til arbeidslager.* KLIBAS-report no. 05/94, DNMI, Oslo, 1994.
- [4] Øgland, P. *Omlegging av databaserutiner ved overgang fra Oracle6 til Oracle7.* KLIBAS-report no. 28/94, DNMI, Oslo,

In addition to these six groups of files there is also a group called the *STAT files* that is presently not being used by the system, but was designed for producing statistics concerning how long the program had to wait for a particular station to report an observations. The purpose of the files was in other words to make a list over stations order by how punctual each station was. As this has turned out to be a less important issue than originally thought, the functions in SYNO\_INN used for reading and updating these files have been temporarily put out of function.

- 1994.
- [5] Øgland, P. *Innlasting av synoptiske data til arbeidslager. Revidert utgave.* KLIBAS-report no. 40/94, DNMI, Oslo, 1994.
  - [6] Øgland, P. *Innlasting av synoptiske data til arbeidslager. Versjon 2.0.* KLIBAS-report no. 18/95, DNMI, Oslo, 1995.
  - [7] Øgland, P. *Driftsrapport juni 1995.* KLIBAS-note no. 09/95, DNMI, Oslo, 1994.
  - [8] Schøyen, A. *Bruerveiledning LOG-DATABASE V2.0.* DNMI, EDB-avdelingen, Oslo, 1994.
  - [9] Øgland, P. *Eksperimentell overføring av data fra syno-filer til Oracle-database. Versjon 3.0.* KLIBAS-report no. 15/96, DNMI, Oslo, 1996.
  - [10] Øgland, P. *Overføring av data fra syno-filer til tabeller SYNOP og TELE. Versjon 3.1.* KLIBAS-report no. 04/97, DNMI, Oslo, 1997.
  - [11] Øgland, P. *Dataoverføring SYNO\_INN v.3.2 fra syno-filer til tabeller SYNOP og TELE med utvidet sikkerhet.* KLIBAS-report no. 55/97, DNMI, Oslo, 1997.
  - [12] Øgland, P. *SYNO\_INN v.3.3: Revised for inserting international synops into TELE.* KLIBAS-report no. 03/98, DNMI, Oslo, 1998.
  - [13] Øgland, P. *Reading data from syno-files into KLIBAS: SYNO\_INN v.3.4.* KLIBAS-report no. 54/98, DNMI, Oslo, 1998.
  - [14] Øgland, P. *Reading data from syno-files into KLIBAS: SYNO\_INN v.3.5.* KLIBAS-report no. 59/98, DNMI, Oslo, 1998.
  - [15] Øgland, P. *Reading data from syno-files into KLIBAS: SYNO\_INN v.3.6.* KLIBAS-report no. 65/98, DNMI, Oslo, 1998.

## Reading PIO observations into the KLIBAS database system at DNMI

Petter Øgland

Norwegian Meteorological Institute  
December 4th, 1998

### ABSTRACT

In order to have a reliable routine for reading PIO observations (observations generated by use of Personal Computers on the observation sites) into the KLIBAS database system at the Climatology Division at DNMI, the PIO\_INN program has been continuously revised during the second half of 1998. The system is still, however, not too stable, and while the reason for the frequent collaps is not fully understood to the extent of having eliminated all problems, the latest reason for the system breaking down is discussed and analysed in this paper, suggesting reprogramming that may make it more robust.

### PIO weather stations

Primo December 1998, the PIO\_INN computer program is reading observations from eight PIO weather stations. Figure 1 shows the number of files read each month since the initiation of the system in Mars 1998.

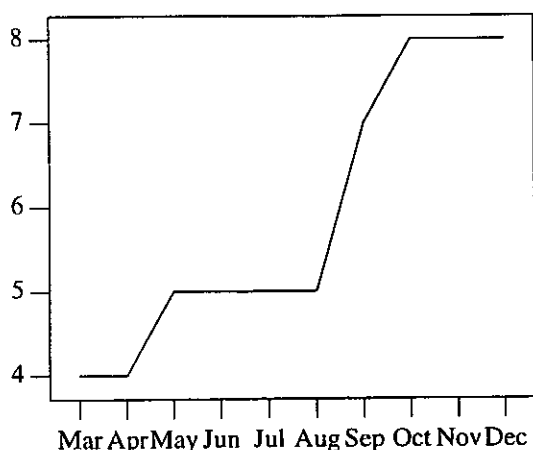


Fig 1. Number of files with PIO observations

On the area where the PIO files are to be found, observation files from semi-automatic weather stations (SAWS) are also placed. The first test SAWS being from June 1998. In July there were 9 stations of this kind, in August to November 14 while there have been none in December so far.

### Program development

The development of the PIO\_INN system has so far consisted of two phases. Shortly after PIO data were a part of the DNMI dataflow systems on Mars 23rd 1998, see user guide in [1], an initial version of the PIO\_INN program, described in [2], was operative in the sense that it was reading observations from the pio-files into the PIO datatable in the Oracle RDBS of the KLIBAS database system on a daily basis. An instruction on how the Climatology Division were to handle PIO observations is described in [3].

The files generated during these early stages of the PIO project contained only parameters for barometre temperature (Bp), air pressure (P0, PF, PT), evaporation (EV), air temperature (TT, TnT, TN\_12, TxT, TX\_12), ground level temperature (TG\_12), water temperature (TW) and relative humidity (UU), all acronyms explained in [1].

The initial version of PIO\_INN was revised in July 1998 by adding log functions in order to add a systematic control on whether the program was running according to specifications or not and to which extent observations were being made at correct time. The version 1.1 of the system, that is described in [4], was also augmented in order to handle a complete set of parameters, as described in [1], not only the test parameters being used on early files.

During August 1998 the PIO\_INN system was significantly reprogrammed in order to merge data from semi-automatic weather stations (SAWS) into the PIO dataflow as documented in [5]. Due to the complex nature of the SAWS files, using a mixed approach for marking missing values, including attainable values to signify missing ones, this second version of the PIO\_INN system has been so far revised twice, documented in [6] and [7].

In the version 2.1 of PIO\_INN, described in [6], statistical charts were added to be produced by the program in order to make it easier to see whether the program was performing normally or not. Problems having to do with data format on the SAWS files was systematically documented and reported to those responsible for the producing the files.

The version 2.2, described in [7], went a step further by adding a simple quality control routine to the program in order to eliminate observations that could not be inserted into the PIO datatable, causing the system to break down, and displaying quality control results by methods of statistical process control (SPC) as a help to detect whether the program was under control or not.

### Robustness

The first version of PIO\_INN started running on the 12th of May 1998. In figure 2 the relative number of abnormal (defective) terminations of the program so far is plotted.

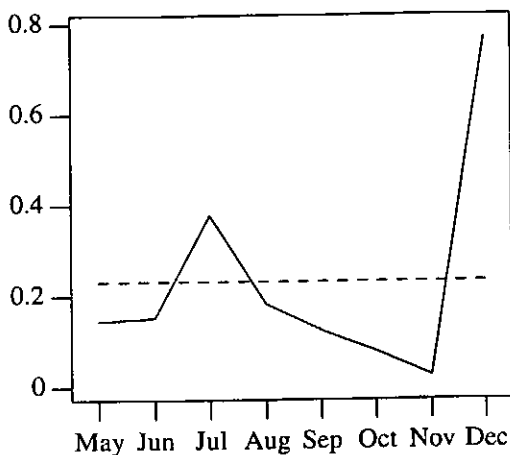


Fig 2. Relative number of executions to fail 1998

As can be seen in figure 2, with exception of recent program failure of early December, the tendency since the design of version 1.1 in July

1998 has been a gradual improvement of the system in terms of reporting less and less defects on the average each month. The general average relative number of defects for the eight month period is 0.231 as indicated by the dashed line.

The program is run about 200 times each month on the average. According to automatic runs by the crontab schedule where it is to be executed every 3 hours or eight times a day, it should run 240 times on a 30 day month. The average being below this value is explained, however, by the three hour schedule run by the crontab prior to redefinitions in October. IOn the beginning the program was automatically run only four times a day.

### The December problem

In addition to logging the number of failures each month, the program also logs the cause of failure, meaning, of course, that the only defects used in these statistics are the ones that are being detected by the system itself. Program crash because of bus error or other technical fault causing uncontrolled abortion are not normally logged.

There are two types of problems that are being logged by the system. The first one is called warnings and it indicates that something is wrong in the system, although not critical enough to stop execution. The other type is called errors, causing immediate controlled breakdown of the program, logging the sequence of functions that resulted in the fatal state.

As identification of errors are kept on an revolving 12 month basis, the reasons for the rise in relative defects level in figure 2 should be found on the log files. In fact, the reason is found by inspection of the log files to be an error having to do with problems reading data from a PIO station at Bråtå - Slettom at Sjøk in Oppland, station no. 15730.

In this case, it can be read from the log files that the error has been reported on October 1st, November 13th, November 20th and then on numerous occasions early December up to present day (December fourth).

It can also be read from the files that the error occurred in the function "insertDefectsStation" which was called by the function "insertData", called by the function "readData", called by the function "system\_test" which was called by the main loop of the program.

By running the program in test mode and using station no. 15730 as test station, the program is constructed in such a manner that the error can be reconstructed. From doing so it is revealed that the program was not designed in a proper manner to handle new stations being added to the string of PIO stations, failing to insert data if the station in question was not already represented in the PIO datatable.

#### Why did the program break down?

Why did it happen that the program only broke down sporadically in October and November while it broke down systematically in December?

The first file for station no. 15730 is the October file. This file only contains one singular observation, the first of October at 11:00 UTC. By using UNIX functions to check file status, we see that the file was last updated on October the first at 12:10 UTC. PIO\_INN being constructed to only read files that have been updated since last session, this file would never be read again.

Similarly the November file was last updated on November 20th at 12:11 UTC, probably meaning that the only updates on this file were on November 13th and November 20th, causing the two breakdowns of the system.

The December file containing observations from the first day at 12:00, 18:00 and 21:00 and then then observations systematically at 00:00, 03:00, 06:00, 12:00, 18:00 and 21:00 UTC is in correspondance with what was noted above.

#### Conclusion

Reprogramming PIO\_INN so that the group of stations being used for checking validity of observations consist of both stations already stored in the PIO table and possible new stations defined by the input files should prevent this type of problems from reoccurring, adding to the stability and robustness of the PIO\_INN system.

#### References

- [1] O. Bonlid, "Inntastingsprogram for Synopstasjoner. Brukerveiledning for Pio versjon 3.2," in *PC i observasjonstjenesten: PIO\_INN v.1.1.* (appendix), KLIBAS-report no. 25/98, DNMI, Oslo, 1998.
- [2] P. Øgland, *Computer program PIO\_INN.* KLIBAS-report no. 19/98, DNMI, Oslo, 1998.
- [3] Å.M. Vidal and P.O. Kjensli, "5 PIO stasjoner ble satt i drift 24. mars" in *PC i observasjonstjenesten: PIO\_INN v.1.1.* (appendix), KLIBAS-report no. 25/98, DNMI, Oslo, 1998.
- [4] P. Øgland, *PC i observasjonstjenesten: PIO\_INN v.1.1.* KLIBAS-report no. 25/98, DNMI, Oslo, 1998.
- [5] P. Øgland, *PC i observasjonstjenesten: PIO\_INN v.2.0.* KLIBAS-report no. 41/98, DNMI, Oslo, 1998.
- [6] P. Øgland, *PC i observasjonstjenesten: PIO\_INN v.2.1.* KLIBAS-report no. 53/98, DNMI, Oslo, 1998.
- [7] P. Øgland, *PC i observasjonstjenesten: PIO\_INN v.2.2.* KLIBAS-report no. 58/98, DNMI, Oslo, 1998.

## Dataflow in the KLIBAS database system at DNMI: Updating SAWS in ALV with AWS observations from ALA

Petter Øgland

Norwegian Meteorological Institute  
December 14th, 1998

### ABSTRACT

Semi-Automatic Weather Stations (SAWS) are handled as a part of the Non-Automatic Weather Stations in the DNMI data preparations routines (ALV) at the Climatology Division. Part of this treatment is an automatic update of values from the Automatic Weather Stations (AWS) that are being treated separately by a routine for exclusively automated weather stations (ALA). A system ALA2ALV is responsible for this update, but even though it has been run systematically since July 1998 it is still not stable, and while the reason for the frequent collaps is not fully understood to the extent of having eliminated all problems, the latest reason for the system breaking down is discussed and analysed in this paper, suggesting reprogramming that may make the program more robust.

### Semi-Automatic Weather Stations

The first Semi-Automatic Weather Station (SAWS) to be established was a station at Jan Mayen on the 4th of October 1995. Since then more and more SAWS have been included in the network. At present there are 16 such stations, as shown on figure 1.

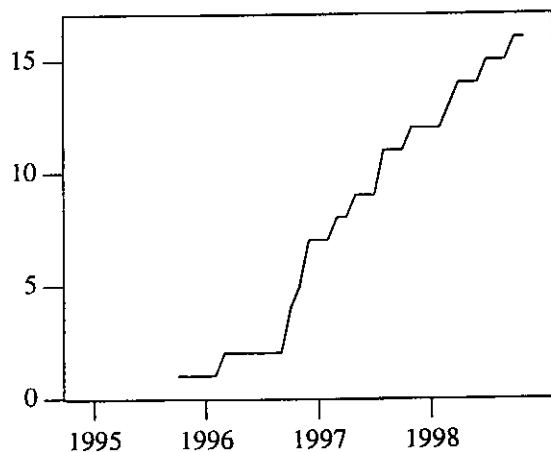


Fig 1. Number of SAWS at DNMI

In December 1995 there was one operative SAWS. In December 1996 this had increased to seven stations, and then again 12 in December

1997 and 16 in December 1998. On the average four stations have been added to the network each year.

### Program development

The development of the ALA2ALV program has so far consisted of two versions, the initial version [1] and a revised version 1.1 documented in [2].

The initial version of the program was made as a test program, and did only update values in the data table TELE. In August 1998, as the program had been running successfully for about a month, the revision 1.1 was established which included functions for updating and flagging observations in the data table ALV.

Updates in the TELE table were flagged with values '5' in order to distinguish the update from other types of updates done by other programs, while a value '1' was used in the ALV table as this table only accepts values '0' and '1'.

### Robustness

The first version of ALA2ALV started running on the 23rd of July 1998. In figure 2 the relative number of abnormal (defective) terminations



of the program so far is plotted.

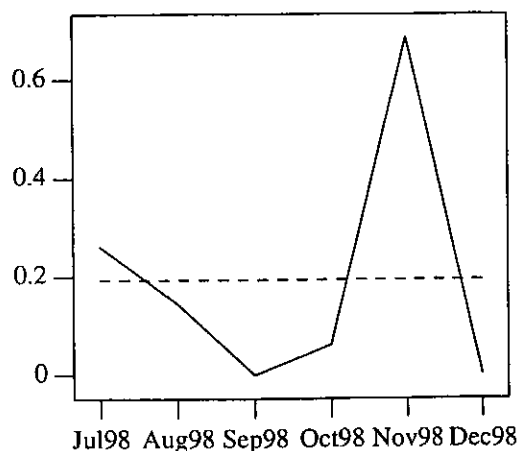


Fig 2. Relative number of executions to fail

As can be seen in figure 2, with exception of program failure in November, the tendency since the design of version 1.1 in August 1998 has been a gradual improvement of the system in terms of reporting less and less defects on the average each month. The general average relative number of defects for the eight month period is 0.1928 as indicated by the dashed line.

The program is run about 85 times each month on the average, including numerous test runs during this period. The program is designed to be run as a part of the AUTO\_INN system, documented in [3], and should hence be run once a day.

Although not indicated on figure 2, the program has been systematically terminating in error since 25th of November and all of December up to the present date.

### The November/December problem

In addition to logging the number of failures each month, the program also logs the cause of failure, meaning, of course, that the only defects used in these statistics are the ones that are being detected by the system itself. Program crash because of bus error or other technical fault causing uncontrolled abortion are not normally logged.

There are two types of problems that are being logged by the system. The first one is called warnings which indicate that something is wrong in the system, although not critical enough to stop execution. The other type is called errors, causing immediate controlled breakdown of the program, logging the sequence of functions that

resulted in the fatal state.

As identification of errors are kept on an revolving 12 month basis, the reasons for the rise in relative defects level in figure 2 in November is to be found on the log files. In fact, the reason is found by inspection of the log files to be an error having to do with problems reading values of PP and TN being out of bounds within a function "sprintfloat".

### Why did the program break down?

One of the reported problems was temperature on Blindern reaching bottom level of 273.1 centigrads on December 10th at 18:00 UTC. Other stations also reported similarly unlikely low temperatures late November and December. The problem is reported to the Instrument Division. Apparently sensors or sensor software may be out of order.

The other problem that is systematically recorded, air pressure tendency of value 5553.5 is registered on a total of 108 observations since November 21st. In this case the IT department is notified if there may have been changes in the observation system using this new value as a "missing value" indicator.

### Conclusion

In respect of how the system has been working, reporting an error is somewhat misleading in this case, and the statement in the sprintfloat function should be adjusted to reporting a warning instead as the problem is handled by the program and does not cause breakdown. The problem has nothing to do with the robustness of the program.

### References

- [1] P. Øgland, *Computer program ALA2ALV*, KLIBAS-report no. 27/98, DNMI, Oslo, 1998.
- [2] P. Øgland, *Updating AWS in ALV and TELE by ALA2ALV v.1.1*, KLIBAS-report no. 45/98, DNMI, Oslo, 1998.
- [3] P. Øgland, *Døgnlige dataoverføring med AUTO\_INN v.1.0 og AUTO2TELE v.2.0*, KLIBAS-report no. 56/98, DNMI, Oslo, 1997.

## Facing a problem of robustness on the MND2HLA AWS data transportation program in the KLIBAS database system at DNMI

Petter Øgland

Norwegian Meteorological Institute  
December 15th, 1998

### ABSTRACT

The program MND2ALA is an interface program that reads observations from a format generated by the AUTO data collection system run by the IT Division and inserts into datatables in the KLIBAS database system run by the Climatology Division. Even though it has been run systematically since October 1995 it is still not completely stable, and while the reason for the frequent collaps is not fully understood to the extent of having eliminated all problems, the latest reason for the system breaking down is discussed and analysed in this paper, suggesting reprogramming that may make the program more robust.

### The MND2HLA program

The MND2HLA program is a part of the AUTO\_INN data transport and storage system that was constructed in order to read AWS data from the "månadsfil"-format, or mnd-format as they are referred to here after, described by Waage in his description of the AUTO data collection and formatting system [1], and inserting observations into Oracle RDMS A-data tables.

The first version of AUTO\_INN, called MND2ALA at the time, was completed in October 1994 [2] and copied only data from the mnd-format files to the ALA data table in the Oracle RDBS KLIBAS. The system was revised in January 1995 (Øgland [3]).

In 1995 the initial programs were augmented into a system named ADI, the first version appearing in Mars 1995 [4] with a revision in May 1995 [5].

The present system, AUTO\_INN was developed from the ADI system in June 1997 [6] and revised in July 1997 [7], and the present MND2HLA program is a part of this system.

### Robustness

With the MND2HLA program there is a log system, updated since October 1995, containing the number of times the program was run, how

many times it failed, how long time it makes on the average on each daily run and how many lines of code the program consists of.

In November 1996 the practice of logging was changed by logging each file instead of the complete session. Recent data is thus plotted in figure 1.

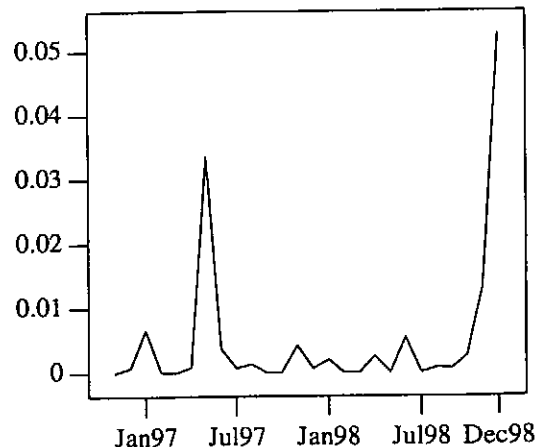


Fig 1. Relative number of executions to fail

As can be seen in figure 1, there is no tendency of the program stabilising. On the contrary, it appears to jump randomly up and down for each consecutive month with particularly nasty cases in May 1997 where 16 of the 476 runs

failed and in November 1998 where 9 of the 681 runs failed causing even more problems in December.

In both these cases the total run of programs is significantly lower than expected. The average number of runs of this program from November 1996 and onwards has been 1160 runs a month. Numbers below this may have to do with the program not being run as the mother program AUTO\_INN has already failed and terminated.

### The November/December problem

In addition to logging the number of failures each month, the program also logs the cause of failure, meaning that the only defects used in these statistics are the ones that are being detected by the system itself. Program-crash because of bus error or other technical fault causing uncontrolled abortion is not normally logged.

There are two types of problems that are being logged by the system. The first one is called warnings and it indicates that something is wrong in the system, although not critical enough to stop execution. The other type is called errors, causing immediate controlled breakdown of the program, logging the sequence of functions that resulted in the fatal state.

As identification of errors are kept on an revolving 12 month basis, the reasons for the rise in relative defects level in figure 2 should be found on the log files. In fact, the reason is found by inspection of the log files to be an error having to do, among other things, with problems reading data from an AWS station at Kvitfjell in Oppland, station no. 13160.

It can also be read from the files that the error occurred in the function "insert\_data\_hla" which was called by the function "insert\_data", called by the function "system\_test" which was called by the main loop of the program.

The Oracle error message in "insert\_data\_hla" function explains that "ORA-01438: value larger than specified precision allows for this column", and by further analysis and debugging of the MND2HLA program it appears the the program is trying to insert values of the number of minutes in an hour when precipitation was measured to be 187 minutes, 202 minutes, 195 minutes and so on but where the column in the data table would only accept two-digit values, preferably between 0 and 60.

### Why did the program break down?

Discussing the problems with the IT Division it appears that the AUTO data collection system was modified around the 25th of November. While the AUTO system normally eliminate values that are far out of range of what would be physically expected, this filter did not function on all sensor elements after modification and hence the breakdown of the MND2HLA program.

### Conclusion

Even though the reason why the MND2HLA program broke down in November 1998 is not very likely to happen again, it was nevertheless a weakness of the MND2HLA program that it did not handle the situation more elegantly. The imperative thing to do then, is to reprogram MND2HLA so that values that wont fit with the table format are automatically rejected with a warning, without causing the whole system to break down.

### References

- [1] Waage, E. *Brukarrettleiing AUTO. Datainnsamling frå automatiske værstasjonar.* KLIBAS-report no. 31/94, DNMI, Oslo, 1994.
- [2] Øgland, P. *Månedlig rutine for innlasting av automatstasjonsdata i arbeidslager* KLIBAS-report no. 32/94, DNMI, Oslo, 1994.
- [3] Øgland, P. *Innlasting og uthenting av automatstasjonsdata. Ny utgave.* KLIBAS-report no. 07/95, DNMI, Oslo, 1995.
- [4] Øgland, P. *ADI: Automatisk datainnlasting for AUTO til arbeidslager* KLIBAS-report no. 13/95, DNMI, Oslo, 1995.
- [5] Øgland, P. *Automatisk datainnlasting for AUTO med månedlig oppdatering av hovedlager: ADI v.1.1* KLIBAS-report no. 16/95, DNMI, Oslo, 1995.
- [6] Øgland, P. *Døgnlign dataoverføring med AUTO\_INN v.1.0 og AUTO2TELE v.2.0* KLIBAS-report no. 56/97, DNMI, Oslo, 1997.
- [7] Øgland, P. *Innlesing AUTO\_INN v.1.1 for AVS: Programmer mnd2ala, mnd2hla, ala2tele, adk og mkk* KLIBAS-report no. 59/97, DNMI, Oslo, 1997.

## Comparing collapse history for the MND2ALA and MND2HLA data transportation program in the KLIBAS database system at DNMI

Petter Øglund

Norwegian Meteorological Institute  
December 16th, 1998

### ABSTRACT

The programs MND2ALA and MND2HLA are interface programs that read observations from a format generated by the AUTO data collection system run by the IT Division and inserting these observations into datatables in the KLIBAS database system run by the Climatology Division. Even though the programs have been run systematically since June and October 1995, they are still not completely stable. While the reason for the frequent collapses are not fully understood to the extent of having eliminated all problems, the latest reason for the programs breaking down is discussed and analysed in this paper, suggesting reprogramming that may make the programs more robust.

### The MND2HLA and MND2ALA programs

Both the MND2HLA and the MND2ALA programs are part of the AUTO\_INN data transport and storage system that was constructed in order to read AWS data from the "månadstil"-format (mnd-format) described by Waage in his description of the AUTO data collection and formatting system [1], and inserting observations into Oracle RDMS data tables.

The first version of AUTO\_INN, called MND2ALA at the time, was completed in October 1994 [2] and copied only data from the mnd-format files to the ALA data table in the Oracle RDBS KLIBAS. The system was revised and re-documented [3] in January 1995.

In 1995 the initial programs were augmented into a system named ADI, the first version appearing in Mars 1995 [4] with a revision in May 1995 [5].

The present system, AUTO\_INN was developed from the ADI system in June 1997 [6] and revised in July 1997 [7], and the present MND2ALA program is a part of this system. The latest version 1.2 of the MND2HLA program is described in [8].

### Collapse history

With both programs there are log systems, updated since June and October 1995 respectively, containing the number of times the program was run, how many times it failed, how long time it makes on the average on each daily run and how many lines of code the program consists of.

In November 1996 the practice of logging was changed for both programs by logging each file instead of the complete session. Only recent data is thus plotted in figures 1 and 2.

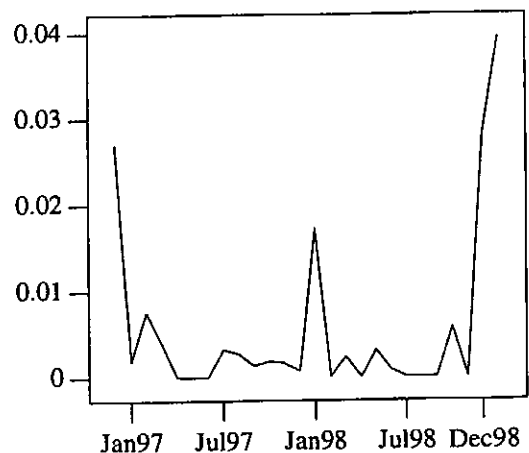


Fig 1. Relative number of failures for MND2ALA

As can be seen in figure 1, there is no tendency of the program MND2ALA stabilising. On the contrary, it appears to jump randomly up and down for each consecutive month with local maxima in December 1996 where 8 out of 1056 runs failed, January 1998 where 4 out of 1699 runs failed and November 1998 where 78 of 1982 runs failed.

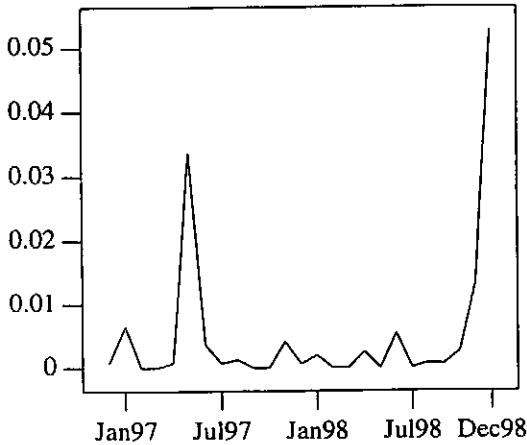


Fig 2. Relative number of failures for MND2HLA

In figure 2 it can be seen that problems in MND2HLA appear equally random as with the MND2ALA program. The extreme cases for this program are recorded in May 1997 where 16 of the 476 runs failed and in November 1998 where 9 of the 681.

In both these cases the total run of MND2HLA is significantly lower than expected. The average number of runs of this program from November 1996 and onwards has been 1160 runs a month. Numbers below this may have to do with the program not being run as the mother program AUTO\_INN has already failed and terminated.

In figure 3 both programs are plotted against each other.

As could also be seen from the separate plots in figures 1 and 2, the scatter plot in figure 3 also shows that there is very little evidence of a linear relationship between the relative number of failures in MND2ALA and MND2HLA.

This loss of linear dependency would indeed be expected as both MND2ALA and MND2HLA are run by the AUTO\_INN system, and whenever MND2ALA breaks down, AUTO\_INN also breaks down and MND2HLA will not be executed.

In other words, the failures recorded by MND2HLA are those that were recorded when MND2ALA was running normally.

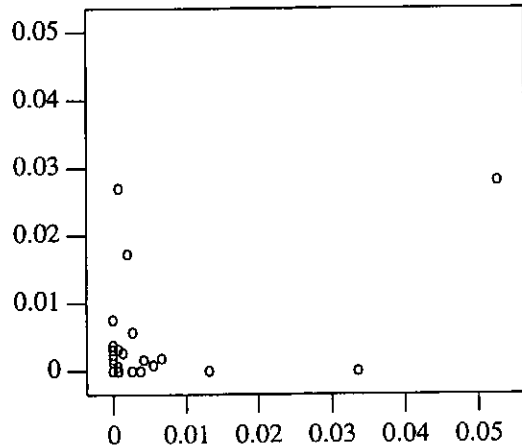


Fig 3. MND2ALA vs. MND2HLA (rel. failure)

When one of the programs MND2ALA or MND2HLA break down, it is often the case that the source code of each program has to be edited, often doing the same type of corrections in parallel.

The reason why the programs are being run separately, the table update not being handled by a single program, is that in the early version of the AUTO\_INN system it was considered sensible to have different parts of the data flow structure work in an orthogonal manner, that is independently of each other.

The system would appear more robust if a problem only caused a part of system to break down, not all of the system. Today, as the run of MND2HLA depend on a normal run of MND2ALA, this argument does no longer seem to hold.

#### Should the two programs be merged into one?

In order to avoid doing something in a rash, a pro-et-contra discussion is presented. Reasons for merging the programs are:

- P1. Even though it was initially a good idea to have MND2ALA and MND2HLA working in parallel, the programs are no longer run in an independent manner. Maintenance of two similarly structured programs is more demanding than just one.
- P2. By reducing the amount of administrative logs the focus on more relevant problems may be higher.

On the other hand, there are also reasons for not merging the programs:

- C1. The programs have been running separately for two and a half year. Unless it is an undemanding task to merge, then system should continue as it is as there are more important issues to address than data flow for AWS.
- C2. It may turn out in the future that the idea of separate flow may be a better one than the integrated approach.

At present, it seems that the pro arguments P1 and P2 are more significant than the arguments C1 and C2. In fact, while the arguments P1 and P2 represent questions that are raised every time the system breaks down, arguments C1 and C2 do not seem very strong at all:

In the case of argument C1, it does not appear likely that the effort of merging the programs will be any more difficult than the more regular effort put in at random times in order to make the system function.

Even though argument C2 may turn out to be right, if, say, MND2HLA is merged into MND2ALA, the programs can still exist in separate code, although MND2HLA will not get updated until it should turn out that argument C2 is valid.

In conclusion, yes, the two programs should be merged into one.

## References

- [1] E. Waage, *Brukarrettleiing AUTO. Datainnsamling frå automatiske v rstationar*, KLIBAS-report no. 31/94, DNMI, Oslo, 1994.
- [2] P.  gland, *M nedlig rutine for innlasting av automatstasjonsdata i arbeidslager*, KLIBAS-report no. 32/94, DNMI, Oslo, 1994.
- [3] P.  gland, *Innlasting og uthenting av automatstasjonsdata. Ny utgave*, KLIBAS-report no. 07/95, DNMI, Oslo, 1995.
- [4] P.  gland, *ADI: Automatisk datainnlasting for AUTO til arbeidslager*, KLIBAS-report no. 13/95, DNMI, Oslo, 1995.
- [5] P.  gland, *Automatisk datainnlasting for AUTO med m nedlig oppdatering av hovedlager: ADI v.1.1*, KLIBAS-report no.

16/95, DNMI, Oslo, 1995.

- [6] P.  gland, *D gkelig dataoverf ring med AUTO\_INN v.1.0 og AUTO2TELE v.2.0*, KLIBAS-report no. 56/97, DNMI, Oslo, 1997.
- [7] P.  gland, *Innlesing AUTO\_INN v.1.1 for AVS: Programmerer mnd2ala, mnd2hla, ala2tele, adk og mkk*, KLIBAS-report no. 59/97, DNMI, Oslo, 1997.
- [8] P.  gland, *Reading AWS mnd-files into A-tables by MND2HLA v.1.2*, KLIBAS-report no. 67/98, DNMI, Oslo, 1998.

## Problems in AWS statistical reports due to running Oracle in different environments for the KLIBAS database system at DNMI

Petter Øgland

Norwegian Meteorological Institute  
December 18th, 1998

### ABSTRACT

Different Oracle environments for different users of the KLIBAS database system at DNMI causes problems when running certain programs. This paper addresses the issue, analysing the cause of problem and makes a suggestion on how to avoid these types of problems.

#### The KLIBAS users and Oracle environment

On the SGI database server (galeha) there are defined 45 users of the "ka" category in the /usr/people directory. For these users the Oracle environment is normally defined on the .bashrc file as default script language is in most cases defined to be the Bourne Again shell (bash) [1].

The AWS statistical reports [2], accessible through the AUTO menu on galeha, were developed by the user kapo, and function normally for this user. For other users, such as kahh, the interface for these programs works less well, giving meaningless symbols as default values.

#### The SQLPATH variable

When running a Sqlplus session, the interface may either run in default mode or it may apply a special mode defined by files that are listed in the SQLPATH environment variable on the .bashrc file.

In this particular case, the user kapo is using the default mode while other users, such as kahh, is using the kapk mode defined on the directory /klima/usr/people/kapk/orasql. In the script login.sql that is executed automatically by all Sqlplus sessions, an 'alter session' command makes the Sqlplus program return confirmative information about the altering of the session.

The way the interface for the AUTO statistics are generated, however, each call to Oracle by the Sqlplus interface does ONLY expect to have search results generated in return, and, in fact,

additional administrative information causes the interface not to function as intended.

#### How to handle the problem

There are at least two ways of handling the problem of unwanted information returned by the Sqlplus interface.

- A1. Altering the .bashrc script for all users.
- A2. Altering the environment locally whenever the programs causing problems are being run.

As alternative A1 would result in a possible update of 44 users, perhaps causing problems in other situations where the information in question may become useful, the alternative A2 is preferable.

It turns out that A2 is not a very difficult way to solve the problem. It can, for instance, be done by adding a statement saying something like `setenv SQLPATH " . : ${HOME} "` in the C-shell environment program for each of the eight programs that currently make up the AUTO menu.

#### References

- [1] S. Parker et al., *UNIX unleashed*, SAMS publishing, Indianapolis, 1994.
- [2] P. Øgland, *Utskriftsrammer for verifikasjon og testutskrift av AVS-data*, KLIBAS-report no. 16/96, DNMI, Oslo, 1996.



## Deciding the future of the CONTSYN1 data check of the KLIMA control routine at the Climatology Division at DNMI

Petter Øgland

Norwegian Meteorological Institute  
December 21st, 1998

### ABSTRACT

On the 16th of November 1998 it was reported that the CONTSYN1 quality check program was not working as specified when handling a semi-automatic weather station at Kjevik, station no. 39040. This particular problem has been analysed, and viable solutions for further development of the CONTSYN1 program is discussed.

### Evolution of the CONTSYN1 program

The CONTSYN1 program is a missing value test which is a part of the KLIMA quality control routine first described in [1] as it was working on the ND-computers and later implemented on the SGI computers in 1995 as described in [2].

The purpose of the CONTSYN1 program, according to [2], is to pin-point situations where either of the elements TT, UU, DD, FF, FB, FX, N, P0, A, PP, S, FG, WW, W1, W2, TN, TX, TG, TW. Cases of EM = 0, H is missing and NH < 0 and the number of days with RR > 0 are also noticed.

The program works as a supplement to CONTSYN2 that also lists missing values. CONTSYN1 is only run by the person in charge of the KLIMA routine, however, and problems spotted by the program usually result in one of the following two cases:

- C1. An observation has been punched in the wrong column or for a wrong date. The value is moved to the right place.
- C2. The observation is missing in the database and also missing on paper. The interpolation expert is then notified in order to fill in an appropriate value.

In 1996 the CONTSYN programs were improved as a version 1.1 of the system became available [3]. The update 1.1 of the CONTSYN

system included a revision 1.1 of CONTSYN1, and although CONTSYN1 was one of the major reasons for updating the system, [3] does not include system code for CONTSYN1 or more specific information on particular problems with the earlier version.

A further update v.2.0 was made of CONTSYN1, documented in [4], but this version contained errors that were difficult to correct, and the revised version was not used after a short initial period.

In July 1998 work on how to redesign and improve the KLIMA quality control routine commenced, and a program KLIMA\_KONTR running the CONTSYN programs in simulation mode was established. The system is described in [5].

### Problems with CONTSYN1

The problems reported on the 16th of November was that the program does not operate properly with semi-automatic weather stations (SAWS). As a consequence of this, and a consequence of how the program is constructed in general, the output generated by CONTSYN1 is enormous. For the SAWS there are several problems.

- P1. Due to unknown reason, the program states that the observation at 06:00 UTC is missing when this is not the case.
- P2. The program reports missing N, VV, WW, W1, W2, NH at 00:00 UTC, which is

indeed the case, but should not be reported as no such observations are done on an SAWS.

- P3. The program reports missing TN, TX and H for cases when these observations are not missing.

Another problem with the present CONTSYN1 program is that it does not produce error logs and run statistics. For most programs in the KLIBAS system, these logs are used for program performance analysis and error detection. At the present, problems with CONTSYN1 can only be detected by manual inspection.

### The cost of maintenance

As explained in [2], all the CONSYN programs on SGI were created by use of common function libraries, depending upon each other in such a manner that compilation of a particular program includes reading of several files that are used by all.

In 1996, however, one of the common files was changed in such a manner that none of the programs can be maintained and compiled. The source code for the programs is now stored in the directory /usr/people/kapo/klima/contsyn.

Even if it were possible to change the code of the original CONTSYN1, at the moment it seems more reasonable to reprogram.

### Further development

There are no difficult algorithms in CONTSYN1, and by keeping a similar interface as was previously used, a new program should be able to fit into the system with less effort than reworking the old.

### References

- [1] P. Øgland, *Kvalitetskontroll av værstasjonsdata i Klimaavdelingen*, KLIBAS-report no. 23/94, DNMI, Oslo, 1994.
- [2] P. Øgland, *Programvare for kvalitetskontroll av klimadata*, KLIBAS-report no. 29/95, DNMI, Oslo, 1995.
- [3] P. Øgland, *Rutine for kvalitetskontroll av klimadata. Versjon 1.1*, KLIBAS-report no. 10/96, DNMI, Oslo, 1996.
- [4] P. Øgland, *Upgrading of the Contsyn System for Verification of Linke Data. Contsyn v2.0*, KLIBAS-report no. 12/96, DNMI,

Oslo, 1996.

- [5] P. Øgland, *KLIMA\_KONTR: Simulation and control of a quality control system for weather data*, KLIBAS-report no. 24/98, DNMI, Oslo, 1998.

## Reading and converting SAWS weather observations V1/V2/V3 into the KLIBAS database system at DNMI

Petter Øgland

Norwegian Meteorological Institute  
December 29th, 1998

### ABSTRACT

In order to have a reliable routine for reading PIO observations (observations generated by use of Personal Computers on the observation sites) and SAWS observations (semi-automatic weather stations) into the KLIBAS database system at the Climatology Division at DNMI, the PIO\_INN program has been continuously revised during the second half of 1998. The system is still, however, not too stable, and while the reason for the frequent collapse is not fully understood to the extent of having eliminated all problems, the latest reason for the system breaking down is discussed and analysed in this paper, suggesting reprogramming that may make it more robust.

### SAWS weather stations

Ultimo December 1998, the PIO\_INN computer program is reading observations from eight PIO weather stations and 14 SAWS. Figure 1 shows the number of files read each month since the initiation of the system in Mars 1998.

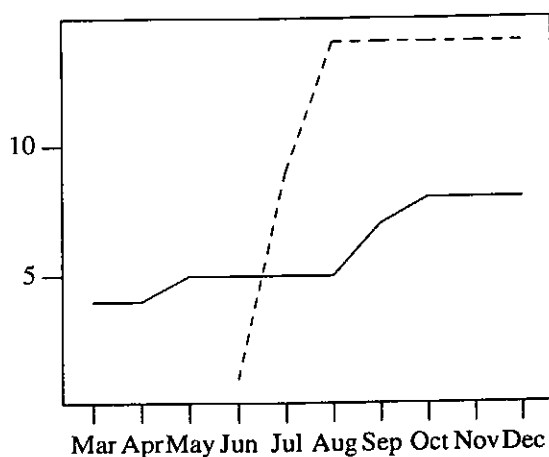


Fig 1. Number of stations: PIO=solid, SAWS=dashed

On the directory where the PIO files are to be found, observation files from semi-automatic weather stations (SAWS) are also placed. The first test SAWS being from June 1998. In July there were 9 stations of this kind, and in August to December there have been 14 stations.

### Program development

The development of the PIO\_INN system has so far consisted of two phases. Shortly after PIO data were a part of the DNMI dataflow systems on Mars 23rd 1998, see user guide in [1], an initial version of the PIO\_INN program, described in [2], was operative in the sense that it was reading observations from the pio-files into the PIO datatable in the Oracle RDBS of the KLIBAS database system on a daily basis. An instruction on how the Climatology Division were to handle PIO observations is described in [3].

The files generated during these early stages of the PIO project contained only parameters for barometre temperature (Bp), air pressure (P0, PF, PT), evaporation (EV), air temperature (TT, TnT, TN\_12, TxT, TX\_12), ground level temperature (TG\_12), water temperature (TW) and relative humidity (UU), all acronyms explained in [1].

The initial version of PIO\_INN was revised in July 1998 by adding log functions in order to add a systematic control on whether the program was running according to specifications or not and to which extent observations were being made at correct time. The version 1.1 of the system, that is described in [4], was also augmented in order to handle a complete set of parameters, as described in [1], not only the test parameters being used on

early files.

During August 1998 the PIO\_INN system was significantly reprogrammed in order to merge data from semi-automatic weather stations (SAWS) into the PIO dataflow as documented in [5]. Due to the complex nature of the SAWS files, using a mixed approach for marking missing values, including attainable values to signify missing ones, this second version of the PIO\_INN system has been so far revised twice, documented in [6] and [7].

In the version 2.1 of PIO\_INN, described in [6], statistical charts were added to be produced by the program in order to make it easier to see whether the program was performing normally or not. Problems having to do with data format on the SAWS files was systematically documented and reported to those responsible for the producing the files.

The version 2.2, described in [7], went a step further by adding a simple quality control routine to the program in order to eliminate observations that could not be inserted into the PIO datatable, causing the system to break down, and displaying quality control results by methods of statistical process control (SPC) as a help to detect whether the program was under control or not.

A breakdown of the system in early December 1998 was analysed and improved according to suggestions made in [8].

### Robustness

The first version of PIO\_INN started running on the 12th of May 1998. In figure 2 the relative number of abnormal (defective) terminations of the program so far is plotted.

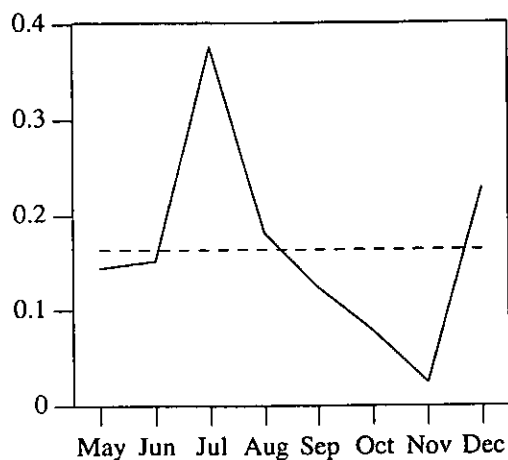


Fig 2. Relative number of executions to fail 1998

As can be seen in figure 2, with exception of recent program failure of early December, the tendency since the design of version 1.1 in July 1998 has been a gradual improvement of the system in terms of reporting less and less defects on the average each month. The general average relative number of defects for the eight month period is 0.163 as indicated by the dashed line.

The program is run about 200 times each month on the average. According to automatic runs by the crontab schedule where it is to be executed every 3 hours or eight times a day, it should run 240 times on a 30 day month. The average being below this value is explained, however, by the three hour schedule run by the crontab prior to redefinitions in October. In the beginning the program was automatically run only four times a day.

### The December problem

In addition to logging the number of failures each month, the program also logs the cause of failure, meaning that the only defects used in these statistics are the ones that are being detected by the system itself. Program crash because of bus error or other technical fault causing uncontrolled abortion are not normally logged.

There are two types of problems that are being logged by the system. The first one is called warnings and it indicates that something is wrong in the system, although not critical enough to stop execution. The other type is called errors, causing immediate controlled breakdown of the program, logging the sequence of functions that resulted in the fatal state.

As identification of errors are kept on an revolving 12 month basis, the reasons for the rise in relative defects level in figure 2 should be found on the log files. In fact, the reason is found by inspection of the log files to be an error having to do with problems reading data from a PIO station at Hekkingen Fyr at Lenvik in Troms, station no. 88690.

In this case, it can be read from the log files that the error has been reported on December 25th, 26th, 27th, 28th and 29th (present day).

It can also be read from the files that the error was of type "ORA-01438: value larger than specified precision allows for this column" when the program tried to insert values of 102 for the column V2 that can only store values between -99 and 99, that is integer values of no more than two digits.

The error occurred in the function "insert-Data" which was called by the function "read-Data", called by the function "system\_test" which was called by the main loop of the program.

By running the program in test mode and using station no. 88690 as test station, the program is constructed in such a manner that the error can be reconstructed.

From doing so it becomes clear that the problems arises from the way the column `_VT_new1` is interpreted and read into variables V1, V2 and V3. The program should not have crashed the way it did, preventing observations entering the PIO table, but rather generated a warning and continued business as usual.

#### Why did the program break down?

On the 25th of December, 12:00 UTC, the `_VT_new1` column contained the value 10240.00. According to specifications in [9], this column should only contain four digits, the two first being V1 and the last two being V2. The value on December 25 violates this.

There seems to be no obvious reason why the column contained a five digit integer, so the only thing to do was to send a message is sent to the IT Division if they could possibly explain what was going on.

#### Conclusion

The program `PIO_INN` should be altered so that it does not try to convert five digit values of `_VT_new1`, `_VT_new2`, `_VT_old1` or `_VT_old2` into weather code. Such instances should be reported as warnings, but otherwise the program should continue normally.

#### References

- [1] O. Bonlid, "Inntastingsprogram for Synop-stasjoner. Brukerveiledning for Pio versjon 3.2," in *PC i observasjonstjenesten: PIO\_INN v.1.1.* (appendix), KLIBAS-report no. 25/98, DNMI, Oslo, 1998.
- [2] P. Øgland, *Computer program PIO\_INN.* KLIBAS-report no. 19/98, DNMI, Oslo, 1998.
- [3] Å.M. Vidal and P.O. Kjensli, "5 PIO stasjoner ble satt i drift 24. mars" in *PC i observasjonstjenesten: PIO\_INN v.1.1.* (appendix), KLIBAS-report no. 25/98, DNMI, Oslo, 1998.
- [4] P. Øgland, *PC i observasjonstjenesten: PIO\_INN v.1.1.* KLIBAS-report no. 25/98, DNMI, Oslo, 1998.
- [5] P. Øgland, *PC i observasjonstjenesten: PIO\_INN v.2.0.* KLIBAS-report no. 41/98, DNMI, Oslo, 1998.
- [6] P. Øgland, *PC i observasjonstjenesten: PIO\_INN v.2.1.* KLIBAS-report no. 53/98, DNMI, Oslo, 1998.
- [7] P. Øgland, *PC i observasjonstjenesten: PIO\_INN v.2.2.* KLIBAS-report no. 58/98, DNMI, Oslo, 1998.
- [8] P. Øgland, Reading PIO observations into the KLIBAS database system at DNMI, *preprint (to appear in the KLIMA report series)*, DNMI, Oslo, 1998.
- [9] H. Østby et al., Navnekonvensjon for måleparametre fra automatstasjoner, manuelle observasjoner, eller en kombinasjon av disse, *preprint*, November 6, DNMI, Oslo, 1998.