Airborne Data Processing and Analysis Software Package

David J. Delene[1]

[1]Department of Atmospheric Sciences
University of North Dakota
Clifford Hall 420
4149 University Avenue
Grand Forks, ND 58202-9006

Correspondance Email: delene@aero.und.edu

Software Article submitted to Earth Science Informatics on 28 December 2009.

Revised and resubmitted to Earth Science Informatics on 4 June 2010.

Keywords: Time Series Measurement, Open Source Scientific Software, Airborne Measurements, Research Aircraft

Abstract The practice of conducting quality control and quality assurance in the construction of data sets is often an overlooked and underestimated task of many research projects in the Earth Sciences. The development of software to effectively process and quickly analyze measurements is a critical aspect of a research project. An evolutionary approach has been used at the University of North Dakota to develop and implement software to process and analyze airborne measurements. Development over the past eight years has resulted in a collection of software named the Airborne Data Processing and Analysis (ADPAA) package which has been published as an open source project on Source Forge. The ADPAA package is intended to fully automate data processing while incorporating the concept of missing value codes and levels of data processing. At each data level, ADPAA utilizes a standard ASCII file format to store measurements from individual instruments into separate files. After all data levels have been processed, a summary file containing parameters of scientific interest for the field project is created for each aircraft flight. All project information is organized into a standard directory structure. ADPAA contains several tools that facilitate quality control procedures conducted on instruments during field projects and laboratory testing. Each quality control procedure is designed to ensure proper instrument performance and hence the validity of the instrument's measurement. Data processing by ADPAA allows edit files to be created that are automatically used to insert missing value codes into a time period that had instrument problems. The creation of edit files is typically done after the completion of a field project when scientists are performing quality assurance of the data set. Since data processing is automatic, preliminary data can be created and analyzed within hours of an aircraft flight and a complete field project data set can be reprocessed many times during the quality assurance process. Once a final data set has been created, ADPAA provides several tools for visualization and analysis. In addition to aircraft data, ADPAA can be used on any data set that is based on time series measurements. The concepts illustrated by ADPAA and components of ADPAA, such as the Cplot visualization tool, are applicable to areas of Earth Science that work with time series measurements.

Introduction

With current electronic instruments and computer technology, the amount of recorded observations available in Earth Science fields, such as Atmospheric Science, is enormous. Recorded observations are generally referred to as data, which by itself does not result in scientific progress but has to be analyzed to extract information and scientific understanding. Pinch (1985) proposed the concept of 'externalization of observation' to point out how a long chain of interpretations are used when reporting on modern scientific observations. In this respect, modern airborne observations made in the atmospheric sciences are no different than observations of solar neutrinos made in astronomy. Both types of experiential observations do not depend much on sense perceptions, but rather on the reliability of the practices and assumptions that went into the observation process. What is important is not what the experimenter 'saw' but rather how carefully the practices were followed and how good the software programs were written. In this paper, I report on the current practices followed, and software used, when constructing airborne data sets in the Department of Atmospheric Sciences at the University of North Dakota. These practices and the software tools are similar to the Federal Reference Method for $PM_{2.5}$ aerosol sampling used by the US Environmental Protection Agency (Noble et al. 2001), in that they both describe practices that need to be conducted and tools used when making measurements that allow for comparisons between data sets. In the case of the airborne measurement described here, the practices and tools extend be on instrumentation to include software programs and data processing methods.

The University of North Dakota has owned and operated a research aircraft since the mid 1970s which has been used to make airborne measurements in many government funded field projects and privately sponsored programs. Measurements from the University of North Dakota’s Citation Research Aircraft has resulted in many scientific publications (e.g. Prenni et al. 2007; Sukovich et al. 2009) and contributed to the development of airborne systems such as the Tropospheric Airborne Meteorological Data and Reporting (TAMDAR) system (Murray et al. 2005). Recently, a new Cessna Citation II jet aircraft has been purchased and modified to carry instruments to perform airborne research. All instruments provide some type of measurement that needs to be recorded for further scientific analysis. Most modern instruments provide either an on-board data recording method or stand alone data recording software. On an observational platform such as an aircraft, it is desirable to have a central system that acquires and records all measurement in real time. A single central data recording system ensures that all measurements are time

synced and that only one data file needs to be archived for each aircraft flight.

Central aircraft data acquisition systems are commercially available, or can be custom made. Custom made aircraft acquisition systems have the disadvantage of typically being poorly documented and only a small group of people, sometimes only one person, understands how to configure the system. Commercial systems vendors, on the other hand, can spread the development and documentation cost across several research groups and hence provides better documentation and more features at lower cost relative to one-of-a-kind custom systems.

The University of North Dakota owns and operates the Science Engineering Associates, Inc. model M300 Data Acquisition System (M300) for acquiring and recording airborne measurements (Science Engineering Associates 2009). The M300 (Fig. 1) is built on the QNX real-time operating system and is capable of synchronous and asynchronous data acquisition. The system is configurable via a graphical interface with setup information saved in editable tables. The M300 has real-time data processing functionality that is utilized to display measurements to on board personnel and for data down linking.

A software environment for post-processing data recorded by a data acquisition system, such as the M300, should have the following characteristics:

- A programming environment that enables rapid software development;
- A flexible, non-proprietary software environment;
- A robust environment for testing processing software;
- A modular data processing environment;
- The ability to use open-source code;
- A method to simply and completely automate data reprocessing; and
- The ability to work with a complete flight data set at one time so measurements at a future point in time can be used in a data processing algorithm.

While there has been a lot of progress in instrumentation and computer hardware used in Earth Science research over the past several decades, it is only recently that software to process and analyze the vast amount of data recorded by data acquisition systems from many different types of instruments has been developed. The necessity to work with data from many different instruments and the lack of robust software tools has lead scientists to develop their own software tool sets. At the University of North Dakota, a modern post-processing method that begins with the binary data file created by the M300 data acquisition system was started from scratch

in 2002 since no software environment existed with the characteristics listed above. Previous to 2002, aircraft data was post-processed using a single large FORTRAN module. The FORTRAN module was time consuming to modify, difficult to understand, and very difficult for multiple developers to work on. Another alternative processing method is real-time data processing within the data acquisition system itself. This type of data processing has the following short comings:

- Data acquisition systems are a difficult and time consuming programming environment.
- Commercially available data acquisition systems are proprietary development environments that do not use open source software.
- There is a limited ability to test processing software since data acquisition systems are specialized computer systems with limited availability.
- Data acquisition systems are not modular data processing environments.
- Reprocessing of data using a data acquisition system is difficult to automate.
- There is absolutely no way of using measurements at a future point in time in a data processing algorithm when processing data in real time.

With the realization that no truly similar software environment existed (Holzwarth et al. 2010) and to facilitate future development outside the airborne research community, an open source project, Airborne Data Processing and Analysis (ADPAA), was started on 8 November 2008 (Source Forge 2009). Using open source software allows others to examine the processing methodology implementation, to facilitate learning of software development techniques, and to collaborate closely with other scientists on processing algorithm development, and is not restricted to one vendor’s software solution. Likewise, those scientists who use close source processing software, in either instruments or data acquisition systems, are hindered in trouble shooting problems and there is no easy way for outside scientific review of an algorithm’s implementation. The increasing complexity of many modern processing algorithms has rendered “black box” testing (validating the output given an input) insufficient to validate the implementation of an algorithm. The ability to conduct “white box” testing (examination of the source code) should be a prerequisite for scientific publication of an instrument's measurement.

Design and Implementation

Objectives

The main design objectives of ADPAA are to provide the following:

- A completely automated system that enables preliminary data to be generated shortly after each aircraft flight;
- A system that fully implements the concept of missing value codes;
- Data files that are self documenting using the concept of meta-data;
- Standard directory structure for storing files;
- A system to enable quick analysis of data for quality control purposes;
- Implementation of a quality assurance process whereby manually created edits are documented in an “edit” file and automatically applied to “raw” data files to create a “clean” version of the data file; and
- Source code that is easily available to scientist outside the organization.

Automation

Graphical User Interfaces (GUI) enables easy point and click instructions but do not allow for easy automation since GUIs require manual intervention. In contrast, a command line interface allows easy automation of data processing routines. The Linux operating system is based on the philosophy of command line interfaces that do simple things while being combined to complete complex tasks. Commands typically utilize arguments given after the command name itself to change execution switches and provide necessary input values. ADPAA follows the UNIX/Linux philosophy (Gancarz 2003) by using custom written shell (csh, bash, etc.) scripts that accept arguments to automate the post-processing of raw data files. Use of shell scripts that are executed on a laptop running Linux allows automatic generation of preliminary data files within hours after completion of an aircraft flight. Furthermore, automated processing scripts make it trivial to reprocess data.

The most important design objective of all the automatic processing routines is to have code that is clearly documented and understandable. Documentation includes a meta-data file header and references for all scientific equations implemented by the code. Comprehensible code is considered far more important than execution speed. While some techniques are used to improve code execution speed, such as limiting file input/output by utilizing memory, they are only used when they do not

increase code complexity. ADPAA has followed the agile development philosophy where flexibility and learning are stressed over control and efficiency (Subramanian et al. 2009). Hence, scripts are utilized over a programming environment, like FORTRAN or C, since scripts are more flexible for linking modules together.

To facilitate processing via scripts, the concept of data processing levels, whereby data at higher levels are derived from lower level data, is utilized. The data processing level concept used by ADPAA is similar to data levels used by NASA for remote sensing data products (National Aeronautics and Space Administration 1986). Data level 0 contains the “raw” data files created by a data acquisition system. At data level 1, a data file is created for each aircraft instrument. At data level 2, parameters are converted from engineering units (e.g. volts) to physical units (e.g. Celsius). At data level 3, parameters from different instruments are combined. For example, measurements from a total temperature probe are combined with true air speed measurements to determine ambient temperature. At data level 4, derived parameters from different instruments are combined. For example, the droplet spectrums from a Forward Scattering Spectrometer Probe (FSSP) are combined with the droplet spectrum from a 2-dimensional cloud imaging probe (2DC).

The final step in the data processing methodology is to create a summary file that contains all parameters of scientific interest. The specific parameters contained in a field project's summary file depend on instruments deployed and the project's scientific objective(s). Typically, if a field project is a continuation of an earlier project, the summary file parameters will be the same or very similar (e.g. different type of instrument used to measure the same parameter). The summary data files can contain parameters from any data level. Since parameters of scientific interest can be measured at different frequencies, high frequency parameters are averaged to the summary file frequency. Typically, a frequency of 1 Hz is used for the summary file. Figure 2 illustrates the automatic processing of data that is implemented by ADPAA.

Missing Value Code

Missing value codes are numbers that have a physically unrealistic value, which are used to indicate that a valid measurement is not available. Typically, ADPAA uses a very large number, such as 999999.9999. It is important to note that a value of zero typically can not be used as a missing value code since it can be a valid measurement, such as in the case of an air temperature of 0 ^{0}C.

The concept of missing value codes has been fully incorporated into ADPAA. When processing scripts encounter a missing value code, they do not use the value to calculate the derived parameter but instead substitute

the missing value code of the derived parameter. For example, when ambient air temperature is calculated using the Rosemount Probe temperature and true air speed data, and the true air speed parameter is a missing value code, the ambient air temperature will have its missing value code inserted because a derived parameter can not be calculated if a dependent parameter is missing. Since missing data makes analysis more difficult, there is a tendency to interpolate or use a standard value when a measurement is missing; however, this should not be done when creating a scientific data set. Once the data set has been created, the scientist analyzing the data can decide upon a method (e.g. Healy and Westmacott 1956; Rubin 1976; Horton and Lipsizt 2001) to handle time periods without valid measurements.

Data Format

Standard ASCII data file formatting is used for all data during processing by ADPAA except for the inherently binary image probe data (e.g. 2DC images). While binary data files are smaller, the ease of using ASCII files and the ability to quickly examine values make ASCII files more practical. Furthermore, if file storage size is an issue, ASCII data files can be easily and automatically compressed to binary formats using utilities such as gzip and bzip2.

ADPAA uses a standard ASCII file format based on a NASA specification outlined by Gaines and Hipskind (2009). ADPAA uses a slightly modified version of file format number 1001. The modification is not to restrict the line length to 132 characters but to allow any line length. The 132 character restriction is not required for modern software and having all parameters on a single line enables easy file importing with software such as Perl and Microsoft Excel. The file format includes a meta-data header that fully describes data contained within a respective file, which eliminates the need for users to request such information. In addition, meta-data allows processing scripts and analysis software to be written in a more generic manner. For example, each parameter's missing value code is part of the meta-data so it is not necessary to hard code the missing value code into software. An example ASCII meta-data file with a brief explanation is given in Appendix A.

Project summary data files are created in the standard ASCII format; however, a NetCDF formatted version of summary data files are also created using an automated script. NetCDF files have advantages in that they are standard files which contain meta-data, but they are more difficult to work with and values cannot be viewed using text editors since it is a binary data file format. Also, standard Linux scripts (designed to work on ASCII files) cannot be used on the files. The additional complexity of NetCDF files can make scientific analysis easier, especially if analyzing

multiple flights or data from multiple field projects. However, the additional complexity of NetCDF files is judged to add little value during data processing so ASCII files are used instead of a binary data file format such as NetCDF.

Directory Structure

Using a standard data structure and file names during data processing has advantages similar to using a standard formatted data file. With a standard directory structure, scripts can automatically perform processing tasks on files, such as combining data from all flights into a field project file. In addition to instrument measurement data, the directory structure includes auxiliary data such as flight notes, field problem descriptions, analysis plots, and field project meta-data. Data from other measurement platforms (e.g. surface stations, balloons, satellites) can be included within the directory structure. A detailed description of the directory structure is given in Appendix B. While a formal data base, such as mysql, could be used to store and retrieve information, the additional complexity does not provide sufficient benefits to make it worthwhile. Storing data within a data base does not allow aircraft flight information to be browsed as easily as when it is in a directory structure. Once a data set has been constructed and is ready to be distributed, then a software framework (e.g. Open-source Project for a Network Data Access Protocol (OPeNDAP)) could be used for distribution; however, such a system has not been implemented at this time due to the limited user base. In the future, an effort will be made to implement, alongside the currently implemented directory structure system, a more advanced data set storage system such as OPeNDAP or mysql. Hopefully, this will stimulate users to take advantage of the features that such a system offers. These features will likely be important when conducting analysis across multiple projects.

Results

Quality Control

The terms “quality control” and “quality assurance” have been used in meteorology in many ways (Lee et al. 2004; Heard 2006). In this manuscript, the term “quality control” is used to denote the process of conducting tests to check that measurements are being made correctly and accurately, while the term “quality assurance” is the process of reviewing a data set to eliminate (replace with missing value codes) measurements that are invalid due to known problems.

Quality control of airborne field project measurements typically involves two steps. The first step is to calibrate instruments using a traceable standard, and the second step is to check instrument performance during a

field project. Calibrations should be done in the lab before the start of a field project; however, there are often times when a calibration is done after the conclusion of a field project due to insufficient field project preparation time. ADPAA uses a single library program (Constants) to provide calibration constants. The Constants program requires the measurement date, time, and platform name to provide the correct calibration information for an instrument. The meta-data header in the ADPAA ASCII files contains the measurement date and platform information and is automatically passed into the Constants program during data processing. The M300 data acquisition system is configured to automatically save files with names based on the start date and the project table is configured to include correct values for “Aircraft Type” and “Aircraft ID”. These values are used by the Level 1 processing code to set correct values in the ASCII file meta-data header. Documentation of calibrations for all instruments on a platform is done by defining all calibrations coefficients within a single platform-specific constants subroutine.

The second step in quality control is to check instrument performance during a field project. A performance check can involve conducting a procedure similar to how the instrument is calibrated. For example, the Forward Scattering Spectrometer Probe (FSSP) used to measure cloud droplets is calibrated in terms of droplet size by passing borosilicate glass microspheres of a known size through the instrument. While the FSSP should be calibrated before and after each field project, performance tests that involve passing borosilicate glass microsphere through the probe should be performed on a regular basis during a field project to check that the probe sizing is similar to its calibration. Figure 3 presents results from a series of such FSSP performance checks. While the winter field projects are lower than the theoretical value, they are within the instrument's calibration uncertainty; however, the summer measurements typically have very low values and are not consistent with the instrument's calibration; therefore, measurements during this time period cannot be used for scientific analysis. Dusty optics was probably the cause of the low summer performance check values; however, without conducting performance checks there would be no way of knowing for sure that the FSSP was not performing correctly. Analysis of data from such ill-performing instruments can only lead to incorrect theories and possibly inaccurate conclusions.

In addition to performance checks that involve instrument calibration testing, there are aircraft performance checks to test that an instrument is coupled to the sampling environment. An excellent example where environmental coupling performance checks are critical is an aerosol instrument that uses a pump to draw air into a measurement chamber. For example, when a cloud condensation nuclei (CCN) counter is operated on a pressurized aircraft, a hand operated vacuum pump should be used for a complete test (from inlet to exit) for tubing system leaks. Without

quantitative results from such performance checks, the instrument should not be assumed to be coupled to the environment outside the aircraft. Analysis of data from measurements not completely coupled to the environment of interest cannot result in accurate conclusions about that environment.

While airborne instruments require one or more performance checks to ensure valid data are being obtained, there is only a limited amount of personnel time available during field projects to perform the checks. This limited amount of time underscores the value of software which automatically processes performance check data. ADPAA has several features available to help in conducting performance checks and in quickly reviewing data (Fig. 4). For example, data recorded during FSSP droplet sizing performance checks are automatically processed and can be quickly analyzed using ADPAA's Cplot visualization program. Cplot can be used to determine the exact time interval that microspheres pass through the FSSP, plot the size spectrum, and calculate the average channel value.

The Cplot visualization program is an example of a part of ADPAA that can easily be used for analysis of other time series data sets. Cplot currently can read three different ASCII files formats and is complied as a standalone IDL module that can be executed using the free IDL virtual machine on Linux, Windows and Mac platforms. As a demonstration of the usefulness and robustness of the Cplot routine, a colleague’s time series of surface based measurements of ice nuclei concentrations was plotted with Cplot within minutes of first being shown the data files. For Cplot to be used to analyze a new data set, files have to be in a format that Cplot understands which can at most require writing a file conversion script.

Quality Assurance

The ADPAA Cplot visualization software is designed to assist with quality assurance by quickly providing graphical representation of data for scientific review. Typically, while performing quality assurance, instrument measurements are reviewed for unusual values, instrument auxiliary data are checked, and flight notes consulted for problems. Problems identified are documented in an edit file. The edit file naming convention uses a file name based on an instrument's “raw” data file name, where the “raw” suffix is replaced with an “edits” suffix.

An edit file contains the same meta-data header as a data file with each data line containing a list of parameter values that describe an individual edit to be applied using the format given in Table 1. The parameters document states when to apply the edit, what data parameter to edit, what type of edit to make, when the edit is created, who is making the edit, and why the edit is being made. Currently, the only edit type implemented is “I” which means an “invalid” measurement.

Edit files are automatically detected and processed by ADPAA to create a “clean” (quality assured) file where time periods identified as invalid have “raw” values replaced with missing value codes. The “clean” file is then used for all subsequent data processing, data averaging, and data set analysis. ADPAA's file naming convention uses the “raw” suffix to indicate unedited data at the measurement frequency, the “edits” suffix to indicate a file that documents quantity assurance edits, and the “clean” suffix to indicate a data file that has been quality assured. Additional suffix values (e.g. 10Hz, 1 Hz, 10sec) are used to indicate average data created from a “clean” data file if it exists; otherwise, the averaged data files are created from the “raw” data file.

An example of a data edit of an erroneous spike is given in Fig. 5. The total counts from a Passive Cavity Aerosol Spectrometer Probe (PCASP) range from approximately 500 cm^{-3} to approximately 900 cm^{-3} and then back to 500 cm^{-3} during a three second period. The size spectrum shows that channel one has a very large value compared to channels two and three (right panel of Fig. 5). An example of a valid spike in the aerosol distribution can be seen in Fig. 6. The valid spike is nearly an order of magnitude larger than the surrounding values; however, it lasts for 10 seconds and the size spectrum shows that the spike occurs in several different channels. Since invalid (Fig. 5) and valid (Fig. 6) spikes can both have similar magnitudes, care must be taking when performing quality assurance on PCASP measurements.

Discussion

The ability to quickly process and visualize data, which the ADPAA software package makes possible, enables regularly scheduled performance checks to be conducted throughout a field project. By routinely conducting performance checks, a quality controlled data set is created for scientific analyses. Without robust software like ADPAA, there are typically insufficient personnel resources available on aircraft field projects to perform all required instrument checks to ensure properly functioning instruments. Field scientists do their best job; however, post-project quality assurance often reveals invalid measurements.

The PCASP spike example illustrates by Fig. 5 and Fig. 6 shows the advantage of having an experienced scientist familiar with an instrument’s theory and a respective atmospheric parameter’s normal behavior while conducting data quality assurance. Care must be taken when conducting quality assurance to only make edits that remove truly invalid data. There are certain things to look for with each measurement, but no attempt is made in the ADPAA software package to construct automatic software that applies edits. It would be a very difficult task to write such a program and it is felt worthwhile to have a scientist review all measurements. Development

effort has instead been put into creating software that makes data visualization easy and creation of edit files simple.

As the examples in the previous section illustrated, conducting quality control and quality assurance are important steps to obtaining a valid data set which can be used for scientific analysis; however, an additional step which should be conducted is the timely review of data during field projects. Typically, data is reviewed in real-time by a flight scientist; however, measurements may seem odd but possibly believable. An example of this occurred on the 21 June 2008 Polarimetric Cloud Analysis and Seeding Test 2 (POLCAST2) field project (Fig. 7). Without in-depth post-flight review of the cloud condensation nuclei counter raw measurements, indications of a leak would probably not have been discovered and the rest of the field project's flights could have contained invalid measurements like the 21 June 2008 flight.

White Box Science

The term "White Box Science" is used here analogous to the way white box testing is used in software engineering (Pressman 2005). With White Box Science, scientific results can be checked (tested) not only by comparing scientific results but also by examining, validating and reproducing the internal aspects of the scientific analysis. This concept is analogous to white box testing where the internal working of a software package is tested by examining, validating and testing the source code instead of just conducting black box testing on the results produced given a particular input. As an example, consider measurements from any aircraft instrument used in a scientific field project. With Black Box Science, data processing would be implemented in the instrument itself using an embedded executable to conduct real time processing of measurements to produce derived parameters. With White Box Science, the raw measurements would be recorded and the source codes used to process measurements and calculate derived parameters would be available for scientific review.

White Box science is being practiced when the complete data sets, including raw data files, quality control data (e.g. ground check data and calibration data for an aircraft field project), and data processing source code is published. Some scientists may feel that practicing White Box Science as described in this paper is costly and unnecessary; however, the alternative, Black Box Science, has the real significant cost with respect to one's time and money. Development and deployment of independent instrumentation to enable simultaneous measurements of a parameter is difficult and costly. In addition, when simultaneous measurements are made, and they invariably do not agree, it can be very difficult to know the reasons for disagreement without the ability to examine the processing code

of each instrument.

Black Box processing can be conducted along with White Box Science application. In fact, UND scientists use the M300 data acquisition system to process and display data in real-time on aircraft flights. However, this real-time data is not used for scientific analysis. Many instruments used in scientific research (e.g. Condensation Particle Counter) contain software that only allows Black Box processing. Utilizing these instruments for scientific research is very attractive since their wide user base lowers the instrument's cost. However, for scientific research, these instruments must provide raw measurement parameters, without calibrations being applied, in addition to internally calculated parameters. This allows for development of open data processing software which is critical to scientific research. Ideally, companies would partner with researchers on development of processing software for their instruments. Researchers are interested in accurate measurements and hence will develop processing software that does not utilize as many assumptions. In addition to being an external check on the accuracy of processing software, researchers can be viewed as testers of new methods that the company can implement in their future Black Box data processing. This is similar to the model utilized by the commercial Linux Company, Red Hat Inc., when they started their open source Fedora project.

Conclusion

Software to effectively process and quickly analyze aircraft measurements has been developed. The ADPAA software package can create preliminary data within hours of an aircraft flight and facilitate quality control during field projects. Robust software is critically important to obtaining a scientifically useful data set that can be analyzed to meet project's objectives and effectively move scientific research forward. Without necessary software tools to facilitate quality control and quality assurance of measurements, field projects will only generate meaningless data and not the desired scientific information. Furthermore, data sets that have been quality controlled and quality assured as described in this manuscript, allows for the data sets to be used outside of the research project they were created for and enable the data set to be combined with data sets from other research projects (assuming the other project follow similar procedures). This is very important since research projects in Earth Science do not have enough observations; therefore, being able to combine observations from different projects is very important to address many research questions.

Generation of scientific results in an e-science environment requires attention to data provenance (Simmhan et al. 2005) for which the ADPAA frame work provides a base which is applicable to any times series data set

found in Earth Science. The ADPAA software has been used successfully for many years at the University of North Dakota for processing all Citation Research Aircraft data, in teaching the “Measurement System” graduate course, and for conducting an airborne training program in Riyadh, Saudi Arabia in April 2010. During the UND lead Spring 2009 Field Project in Saudi Arabia, ADPAA was used to process and analyze balloon measurements to make comparisons between aircraft and rawinsonde profiles. During the summer of 2010, ADPAA will be used for processing measurements made on Unmanned Aircraft Systems and surface measurements obtained as part of the POLCAST3 research project. Research projects in Mali and Saudi Arabia lead by the Research Application Laboratory of the National Center of Atmospheric Research used ADPAA software. The ADPAA software package has been demonstrated to several companies (Weather Modification Incorporated, Aventech Research Inc., Airdat LLC). So far, ADPAA has been used mainly for airborne measurements in atmospheric science research programs; however, the software package is able to work with any time series data set. In particular, the Cplot visualization program can be used in the analysis of any time series data set in the Earth Sciences. While the complexity of a programmatic solution, such as ADPAA, may appear to pose a steep learning curve, the amount and complexity of data generated by today's instrumentation are making old solutions such as spreadsheets less efficient in the long-term.

Availability and Requirements

Open Source

In November 2008, an open source project for aircraft data processing was started at Source Forge (Source Forge 2009). Source Forge hosts many open source projects and makes available several facilities, such as version control systems (i.e. Subversion (SVN) and Concurrent Versions System (CVS)), which facilitates open publication of the source code, allows anonymous download of the source code, and enables several code developers to work simultaneously. Furthermore, revisions are date stamped which allows the source code as it existed on a certain date to be downloaded. Original code for a desired file that was used to produce a respective data file can be retrieved and used to reproduce data parameters identical to other completed code executions. This is possible because any file created contains meta-data that lists the respective processing date.

Currently, ADPAA has four developers, two of which began while undergraduate students. With the modular design of ADPAA, undergraduate students can make substantial contribution to research projects by writing code for a particular module without the need to understand either all the

previously written code or the overall processing methodology. The faculty mentor can review and submit code for students (or any new developer) until their coding is mature enough to become a developer themselves. Highly motivated undergraduate students have matured to developer level after working for approximately one year (includes a full summer) on the project. Additionally, the Source Forge project is open to developers outside the University of North Dakota. Current developers are willing to assist new users that are interested in contributing to the project.

ADPAA is licensed using the GNU General Public License, version 3, 29 June 2007 (General Public License, 2009) which allows anyone to use and modify the source code. The GNU General Public License is termed a highly restrictive license since it required that any derivative works remain subject to the same license and by prohibiting the mixing of open and closed source code (Lerner and Tirole 2005). The ADPAA source code can be easily downloaded using a Subversion client (see Software Files section for details). By hosting ADPAA at Source Forge instead of a local university server, it is hoped that a larger number of people will use the software and contribute to the project. A wider user and developer base will likely lead to more robust software while reducing overall development effort.

Operating System and Programming Language

Before development of the current ADPAA processing package was first started back in 2002, the aircraft data processing software was completely implemented using FORTRAN code (SUN compiler) on a UNIX Operating System (OS). To enable code to be developed in less time, it was decided to move towards using the Interactive Data Language (IDL) and C-shell (Csh) wrapper scripts on a UNIX/Linux OS. Presently, in 2010, all the core processing code in ADPAA is implemented in IDL with csh and Bash wrapper scripts and it is typically run on Linux (i.e. Redhat, Fedora, Ubuntu) laptops and workstations. Currently csh scripts are being phased out, with new and revised scripts being written using Bash. In addition to IDL code, ADPAA contains code written in FORTRAN (GNU gfortran compiler), C (GNU gcc compiler), and Perl. While using several programming languages has the disadvantage of varied programming syntax, each language has advantages with respect to ease of code development for particular tasks. For example, Perl is much easier to use than IDL when filtering ASCII data files. Multi-language implementation is essential for an evolutionary software development project where a package has to support ongoing projects (i.e. airborne field projects) while moving toward using new tools (i.e. moving from FORTRAN to IDL and csh to Bash scripts).

Even with the many languages used by ADPAA, it only requires a typical installation of Linux/Unix to run and has been tested on Redhat, Fedora and

Ubuntu. While modification of the IDL source code requires an IDL software license, using ADPAA can be done with only the freely available IDL virtual machine. Data processing is implemented using many scripts (csh, Bash, Perl) and hence requires using a Linux type operating system; however, the Cplot analysis program is written completely in IDL and can be downloaded (Cplot 2010) and run on the Mac and Microsoft Windows series of operating systems using a free IDL virtual machine.

Hardware

The computer hardware requirements of the ADPAA software packages are modest in today's computer environment. Table 2 shows that ADPAA compile time and data set processing time ranges from minor (45 seconds and 100 minutes; respectively) to almost trivial (2 seconds and 34 minutes; respectively). Since ADPAA requires a lot of data input/output, the data set storage location can increase the processing time by 10-20% for data sets stored on a Network File System (NFS) mounted drive and by more 200% for a data set on USB mounted drives (Table 2). Considering that a typical complete airborne data set, POLCAST 2, is approximately 9 Gbytes, the data storage capacity is pretty minor when compared to typical radar or satellite data sets used in Earth Science research.

Future Direction

Current code development is moving away from the use of licensed, commercial products toward the use of open-source software to reduce cost. On the user side, an IDL executable can run using the free IDL virtual machine; however, developers must have access to a paid license. The need for an IDL license could greatly limit the number of potential developers, especially from universities and institutions that are located outside the United States. Furthermore, open-source software solutions have matured to the point of offering those necessary features required for many scientific research projects. Developers of new code are encouraged to use the programming language that they believe will work best, with a preference given to using open-source software. While there are several suitable programming options currently available, it is this author's opinion that Python, with library extension such as NumPy (NumPy 2010) and SciPy (SciPy 2010), is the best overall language for future development. NumPy provides masked array (Masked Array 2010) module to easily work with field data. For two dimensional graphics, matplotlib (Matplotlib 2010) produces publication quality plots including unicode and Latex support and for 3D –VTK-based visualization there is Mayavi (Mayavi 2010). Powerful statistical computing is possible in Python via an R bridge (Rpy 2010) that enables the R programming language to be accessed from within Python code.

Expanding ADPAA to more users and developers will require improved documentation, which is a near term objective. More technical documentation is planned to be added to the ADPAA Source Forge web site. This documentation would include processing diagrams that describe the sequence of data file creation and depict those scripts utilized for processing of data from each respective instrument.

In addition to past and current use of ADPAA for research conducted by the University of North Dakota Citation Research Aircraft, ADPAA has been used with data obtained from the University of Wyoming King Air Research Aircraft and several aircraft operated by Weather Modification Inc. (WMI). While current implementation of ADPAA is set up to process data from a SEA Inc. data acquisition system, it could be easily modified to work with other systems. Furthermore, the basic philosophy of ADPAA is to work with time series data; hence, any time series data could be analyzed using the tools developed in ADPAA. These time series data could be from platforms such as Unmanned Aerial Vehicles (UAV) or from surface sites.

While ADPAA implementation is currently limited to the data collected from instrumentation deployed on previous field projects, and is thus limited to a select user base, the ADPAA analysis tools have a potentially wider user base since they work on any time series data. The main ADPAA analysis tool is Cplot (IDL based) which is used to quickly display customizable data plots. Only limited future development of Cplot is planned in the hope of moving towards a completely new analysis package. Future analysis tool development will look towards utilizing an existing open source (e.g. Python) based project or collaboration on development of a new analysis tool. Since NetCDF is a widely-used format, summary data files are presently converted to NetCDF. However, existing tools (e.g. ncplot, ncview) for analysis of NetCDF files are found to lack in necessary features such as a box and whisker plot option and a statistical data analysis package. A graphical interface driven, open-source software package for visualization and display of time-series and NetCDF-formatted data, that has a large developer base and allows easy addition of new features, would be an ideal analysis tool for our scientific research.

Software Files

The latest version of the Aircraft Data Processing and Analysis (ADPAA) software package can be downloaded from Source Forge using SVN. Note that ADPAA has been moved from using CVS to using SVN as the version control system.

To download ADPAA source code using SVN access:

Create directory /usr/local/ADPAA if necessary

Change ownership if necessary, i.e. chown username /usr/local/ADPAA

```
cd /usr/local/ADPAA
svn co https://adpaa.svn.sourceforge.net/svnroot/adpaa/trunk
cd trunk
mv CVSROOT src .svn ..
cd ..
rmdir trunk
```

Note to compile the many ADPAA modules requires an IDL license.
To build and install executables use the following.

```
Create directory /usr/local/ADPAA/bin
Create directory /usr/local/ADPAA/sav
Create directory /usr/local/ADPAA/share
cd /usr/local/ADPAA/src/build/ && make
```

Define and export variables system (typically in /etc/profile and/or /etc/csh.cshrc).

```
ADPAA_DIR=/usr/local/ADPAA
IDL_PROG=/usr/local/ADPAA/src
SVN_EDITOR=vim
Add /usr/local/ADPAA/bin to your PATH environmental variable.
```

Test the installation by executing a script such as cplot.

Appendix A

ADPAA uses a standard ASCII data file which contains meta-data in the header. An example file is given below. Variable labels in curly brackets are added on the left with an explanation of the labels given below the example data file.

```
21 1001                                          {NLHEAD FFI}
Poellot, Mike                                    {ONAME}
University of North Dakota                       {ORG}
Citation II Aircraft                             {SNAME}
Crystal-FACE Field Project                       {MNAME}
1 1                                              {IVOL NVOL}
2002 07 18 2003 03 28                            {DATE RDATE}
      0.0400                                     {DX}
Time [seconds]; UT seconds from midnight.        {XNAME}
3                                                {NV=Primary Variables}
      1.0000      1.0000      1.0000             {VSCAL=Scale Factors}
 999999.9999 999999.9999 999999.9999             {VMISS=Missing Values}
Static Pressure [mb]                             {VNAME[1]}
Air Temperature from the Rosemount Probe [C]     {VNAME[2]}
Attack Angle Differential Pressure [mb]          {VNAME[3]}
0                                                {NSCOML=Special comment lines}
4                                                {NNCOML=Normal comment lines}
Preliminary Data                                 {DTYPE=Preliminary/Final}
```

```
25 Hz Data                                        {VFREQ=Data Frequency}
 Time         STATIC_PR   AIR_T_ROSE  ATTACK   {VDESC=Short Names}
 s            mb          C           mb       {VUNITS=Parameter Units}
  60082.0000    1017.6173      36.4922      -0.2349 {Parameter line 1}
  60082.0400    1017.6173      36.4957      -0.2349 {Parameter line 2}
  60082.0800    1017.7436      36.4957      -0.2495 {Parameter line 3}
```

NLHEAD: Number of lines (integer) composing the file header. NLHEAD is the first recorded value on the first line of an exchange file.

FFI: ASCII file format number. For the UND Citation aircraft data this will always be 1001.

ONAME: A character string specifying the name(s) of the originator(s) of the exchange file, last name first. On one line and not exceeding 132 characters.

ORG: Character string specifying the organization or affiliation of the originator of the exchange file. Can include address, phone number, email address, etc. On one line and not exceeding 132 characters.

SNAME: A character string specifying the source of the measurements or model results which compose the primary variables, on one line and not exceeding 132 characters. Can include instrument name, measurement platform, etc.

MNAME: A character string specifying the name of the field project that the data were obtained from.

IVOL: Volume number (integer) of the total number of volumes required to store a complete dataset, assuming only one file per volume. To be used in conjunction with NVOL to allow data exchange of large data sets requiring more than one volume of the exchange medium (diskette, etc.).

NVOL: Total number of volumes (integer) required to store the complete dataset, assuming one file per volume. If NVOL>1 then each volume must contain a file header with an incremented value for IVOL, and continue the data records with monotonic independent variable marks.

DATE: UT date at which the data within the exchange file begins. For aircraft data files DATE is the UT date of takeoff. DATE is in the form YYYY MM DD (year, month, day) with each integer value separated by at least one space. For example: 1989 1 16 or 1989 01 16 for 16 January 1989.

RDATE: Date of data reduction or revision, in the same form as DATE.

DX(s): Interval (real) between values of the s-th independent variable, X(i,s), i=1,NX(s); in the same units as specified in XNAME(s). DX(s) is zero for a non-uniform interval. DX(s) is non-zero for a constant interval. If DX(s) is non-zero then it is required that NX(s) = (X(NX(s),s)-X(1,s)) / DX(s) + 1. For some file formats the value of DX also depends on the unbounded independent variable and is expressed as DX(m,s).

XNAME(s): A character string giving the name and/or description of the s-th independent variable, on one line and not exceeding 132 characters. Include units of measure and order the independent variable names such that, when reading primary variables from the data records, the most rapidly varying independent variable is listed first and the most slowly varying independent variable is listed last. Currently this is Time [Seconds] from midnight on day aircraft flight started for all UND exchange files.

NV: Number of primary variables in the exchange file (integer). This number plus one (for the time value) gives the number of parameters in the data file.

VSCAL(n): Scale factor by which one multiplies recorded values of the n-th primary variable to convert them to the units specified in VNAME(n). Currently this is 1 for all UND Citation Aircraft recorded values.

VMISS(n): A quantity indicating missing or erroneous data values for the n-th primary variable. VMISS(n) must be larger than any "good" data value, of the n-th primary variable, recorded in the file. The value of VMISS(n) defined in the file header is the same value that appears in the data records for missing/bad values of V(X,n). Currently the majority of UND parameters use a VMISS value of 999999.9999.

VNAME(n): A character string giving the name and/or description of the n-th primary variable, on one line and not exceeding 132 characters. Include units of measure the data will have after multiplying by the n-th scale factor, VSCAL(n). The order in which the primary variable names are listed in the file header is the same order in which the primary variables are read from the data records, and the same order in which scale factors and missing values for the primary variables are read from the file header records.

NSCOML: Number of special comment lines (integer) within the file header. Special comments are reserved to note special problems or circumstances concerning the data within a specific exchange file so they may easily be found and flagged by those reading the file. If NSCOML=0 then there are no special comment lines.

NNCOML: Number of normal comment lines (integer) within the file header, including blank lines and data column headers, etc. Normal comments are those which apply to all of a particular kind of dataset, and can be used to more completely describe the contents of the file. If NNCOML=0 then there are no normal comment lines.

DTYPE: Version description of the data. Typically either Preliminary or Final Data.

VFREQ: Time frequency of the data.

VDESC: A character string on a single line containing a short description of each variable in the exchange file. No spaces are allowed in each short variable description.

VUNITS: A character string on a single line containing the units of each variable in the exchange file. No spaces are allowed in each unit's description.

Appendix B

The directory structure for ADPAA data sets is described below. The description starts from the top of the directory tree and works downward. Each level is defined by a name, notes, and examples. The name is one or two words that define the name of the level. The notes section contains a short description of the level. The example section contains example directories related to the Saudi Arabia 2007/2008 Winter field project. Items in the directory given below are indented as to indicate which directory level they are contained within. The directory tree used by ADPAA has a general directory structure tree as follows.

Project Name/
General Time Period/
General Data Type/
General Instrument Type/
Measurement Purpose/
Particular Time/
Particular Data Type/

A specific example of the directory tree used by ADPAA is given below with the general directory structure tree name being referenced highlighted in bold fonts.

NAME:

Project Name

NOTES:

This is the top of the directory tree. It groups projects by geographical regions.

EXAMPLES:

SaudiArabia/
Mali/

NAME:

General Time Period

NOTES:

Groups time periods together that span a single deployment or similar atmospheric conditions. All sub-directories will follow a similar structure.

EXAMPLES:

Mali/
SaudiArabia/
    Spring07/
    Winter0708/
    Summer08/

NAME:

General Data Type

NOTES:

Groups different types of data together based on where it is obtained.

EXAMPLES:

Mali/
SaudiArabia/
    Spring07/
    Winter0708/
        Aircraft/
            Directory that contains all aircraft data from the winter project, located in the Winter0708 folder.
        Documents/
            Directory that contains documents created or related to the winter project.
        Forecast
            Directory that contains the forecast data for the winter project. Forecast data is grouped into year, month, day sub-directories.
    Summer08/

NAME:

General Instrument Type

NOTES:
Groups data from different platforms and instruments together.
EXAMPLES:

```
Mali/
SaudiArabia/
        Spring07/
        Winter0708/
                Aircraft/
                        KingAir_N825ST/
                Documents/
                Forecast/
        Summer08/
```

NAME:
Measurement Purpose
NOTES:
Groups data that have a similar purpose together.
EXAMPLES:

```
Mali/
SaudiArabia/
        Spring07/
        Winter0708/
                Aircraft/
                        KingAir_N825ST/
                                DMTCCNCTest/
                                        Contains all test data for the DMT and
                                        CCNC instruments.
                                Documents/
                                        Directory for aircraft specific
                                        documentation.
                                Flight/
                                        Directory for the aircraft flight data.
                                GroundChecks/
                                        Directory for the data related to all
                                        calibration checks and ground tests.
                        Documents/
                        Forecast/
        Summer08/
```

NAME:
Particular Time
NOTES:
Groups flights that have a similar start times together. Directory name based on the start time for the data. If flight has more than 1 files,

directory should be named YYYYMMDD_?, where ? is the number of the flight. Under this directory the names should then be similar to the standard directory.

EXAMPLES:

Mali/
SaudiArabia/
Spring07/
Winter0708/
Aircraft/
KingAir_N825ST/
DMTCCNCTest/
Documents/
Flight/
20080308_074553/
Directory should have a name of YYYYMMDD_HHMMSS. The name is unique. Additional information is given in upper level directory or can be put in a readme file in the directory itself.
GroundChecks/
Documents/
Forecast/
Summer08/

NAME:

Particular Data Type

NOTES:

Groups data together based on particular data type.

EXAMPLES:

Mali/
SaudiArabia/
Spring07/
Winter0708/
Aircraft/
KingAir_N825ST/
DMTCCNCTest/
Documents/
Flight/
20080308_074553/
Combined/
Directory that contains data

from multiple data streams.

Notes/
Directory that contains flight

M300_Tables/
Contains the M300 tables used during the flight.

Photos/
Directory for digital images from the flight.

PostProcessing/
Directory for the post-processing data stream which is based on the *.sea file.

QuickChecks/
Directory for plots of the data.

RealTime/
Directory for the real-time data stream which is based on the

Tamu/
Directory that contains the DMA and DMT CCNC data streams.

Video/
Directory to store any video from the flight.

GroundChecks/
Documents/
Forecast
Summer08/

Acknowledgments Several research projects have indirectly funded the development of the ADPAA software package and several people have helped with the software development. Roelof Burger and Duncan Axisa have helped with the development of the data directory structure. Chris Kruse, David Keith, Gökhan Sever, Fred Remer, Mike Poellot and Aaron Bansemer have reviewed and commented on draft versions of the manuscript.

References

Cplot (2010) Download ADPAA Files Now. https://sourceforge.net/projects/adpaa/files/. Accessed May 2010

Gaines SE, Hipskind RS (2009) Format Specification for Data Exchange. http://aerosol.atmos.und.edu/ADPAA/formatspec.txt. Accessed July 2009

Gancarz M (2003) Linux and UNIX Philosophy, Digital Press, 12 Crosby Drive, Bedford, MA 01730, USA

General Public License (2009) GNU General Public License Version3, 29 June 2007. http://www.gnu.org/copyleft/gpl.html. Accessed August 2009

Healy M, Westmacott M (1956) Missing Values in Experiments Analysed on Automatic Computer, Journal of the Royal Statistical Society, Series C (Applied Statistics) 5(3):203-206

Heard DE (2006) Field Measurements of Atmospheric Composition, in: Analytical techniques for Atmospheric Measurement, Blackwell Publishing, Oxford, UK

Holzwarth S, Freer M, Bachmann M, Wang X (2010) FP7-N6SP-DN6.2.1-List of Existing Data Pre-processing Software. http://www.eufar.net/search/doc/doc_pres.php?id_doc=4343&all=1. Accessed May 2010

Horton NJ, Lipsitz SR (2001) Multiple Imputation in Practice: Comparison of Software Packages for Regression Models with Missing Variables. The American Statistician 55(3):244-254

Lee X, Massman W, Law B (2004) Post-field Data Quality Control, in: Handbook of Micrometeorology: A Guide for Surface Flux Measurement and Analysis, Kluwer Academic Publishers, Dordrecht, Netherlands

Lerner J, Tirole J (2005) The Scope of Open Source Licensing. Journal of Law, Economics, and Organization 21(1):20-56; doi:10.1093/jleo/ewi002

Matplotlib (2010) Python Plotting. http://matplotlib.sourceforge.net/. Accessed May 2010

Masked Arrays (2010) NumPy v1.5.dev8106 Manual (DRAFT). http://docs.scipy.org/doc/numpy/reference/maskedarray.html. Accessed May 2010

Mayavi (2010) 3D Scientific Data Visualization and Plotting. http://code.enthought.com/projects/mayavi/. Accessed May 2010

Murray JJ, Nguyen LA, Daniels TS, Minnis P, Schaffner PR, Cagle MF, Nordeen ML, Wolff, CA, Anderson MV, Mulally DJ, Jensen KR, Grainger CA, Delene DJ (2005) Tropospheric Airborne Meteorological Data and Reporting (TAMDAR) icing sensor performance during the 2003/2004 Alliance Icing Research Study (AIRS II). 43rd AIAA Aerospace Sciences Meeting and Exhibit - Meeting Papers Pages 11935-11945

National Aeronautics and Space Administration (1986) Report of the EOS Data Panel, Vol IIa Earth Observing System Data and Information System. Technical Memorandum 87777, National Aeronautics and Space

Administration (NASA), Washington, DC
Noble CA; Vanderpool RW; Peters TM; McElroy FF; Gemmill DB, Wiener RW (2001) Federal Reference and Equivalent Methods for Measuring Fine Particulate Matter. Aerosol Science and Technology 34(5):457–464
NumPy (2010) Scientific Computing Tools for Python. http://numpy.scipy.org/. Accessed May 2010
Pinch T. (1985) Towards an Analysis of Scientific Observation: The Externality and Evidential Significance of Observational Reports in Physics. Social Studies of Science 15(1):3-36
Prenni AJ, Harrington JY, Tjernstom M, DeMott PJ, Avramov A, Long CN, Kreidenweis SM, Olsson PQ, Verlinde J (2007) Can ice-nucleating aerosols affect arctic seasonal climate? Bulletin of the American Meteorological Society 88(4):541-550; doi:10.1175/BAMS-88-4-541
Pressman RS (2005) Software Testing Techniques, in: Software Engineering: A Practitioner's Approach, 6th ed., McGraw Hill, New York, USA, 389-428
Rpy (2010) Low-level Interface to R. http://rpy.sourceforge.net/rpy2.html. Accesed May 2010
Rubin DB (1976) Noniterative Least Squares Estimates, Standard Errors and F-Tests for Analyses of Variance with Missing Data. Journal of the Royal Statistical Society Series B (Methodological) 38(3):270-274
Science Engineering Associates (2009) M300 Data Acquisition System. http://www.scieng.com/support/m300.htm. Accessed December 2009
SciPy (2010) Open-source software for mathematics, science, and engineering. http://www.scipy.org/. Accessed May 2010
Simmhan YL, Plale B, Gannon D (2005) A survey of data provenance in e-science. SIGMOD Rec. 34(3): 31-36; DOI= http://doi.acm.org/10.1145/1084805.1084812
Source Forge (2009) Airborne Data Processing and Analysis. http://sourceforge.net/projects/adpaa/. Accessed July 2009
Subramanian GH, Gary K, Jiang JJ, Chien-Lung C. (2009) Balancing four factors in system development projects. Communications of the ACM 52(10):118-121
Sukovich EM, Kingsmill DE, Yuter SE (2009) Variability of graupel and snow observed in tropical oceanic convection by aircraft during TRMM KWAJEX. Journal of Applied Meteorology and Climatology 48(2):185-198

Figures and Captions

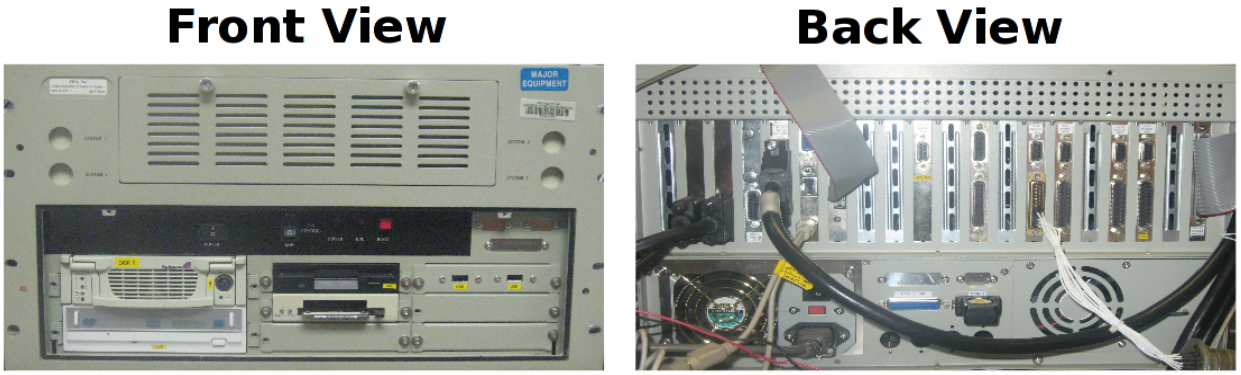

Fig. 1 Science Engineering Associates, Inc. model M300 Data Acquisition System used for aircraft field projects at the University of North Dakota.

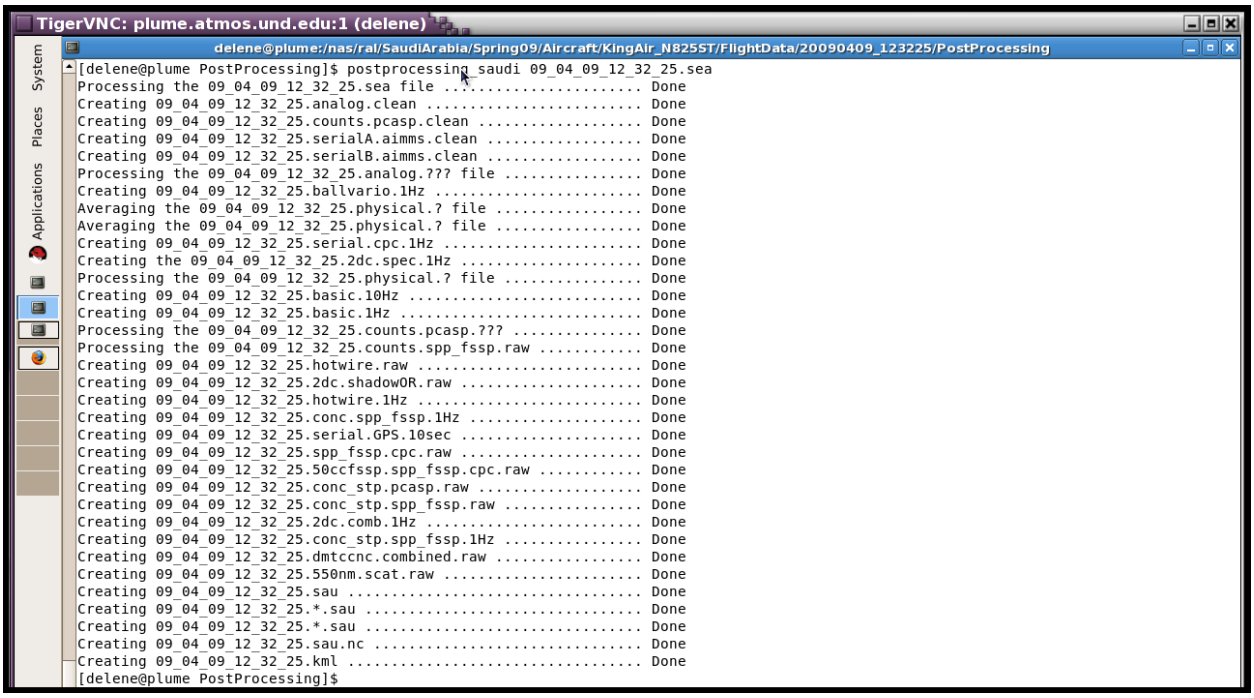

Fig. 2 A screen shot showing the automatic processing of the "09_04_09_12_32_25.sea" data file to obtain a summary file containing parameters of scientific interest. The processing time was 5 minutes 49 seconds on a workstation, 6 minutes 9 seconds on a workstation with a remote data file system (GB network), and 6 minutes 19 seconds on a laptop used in the field.

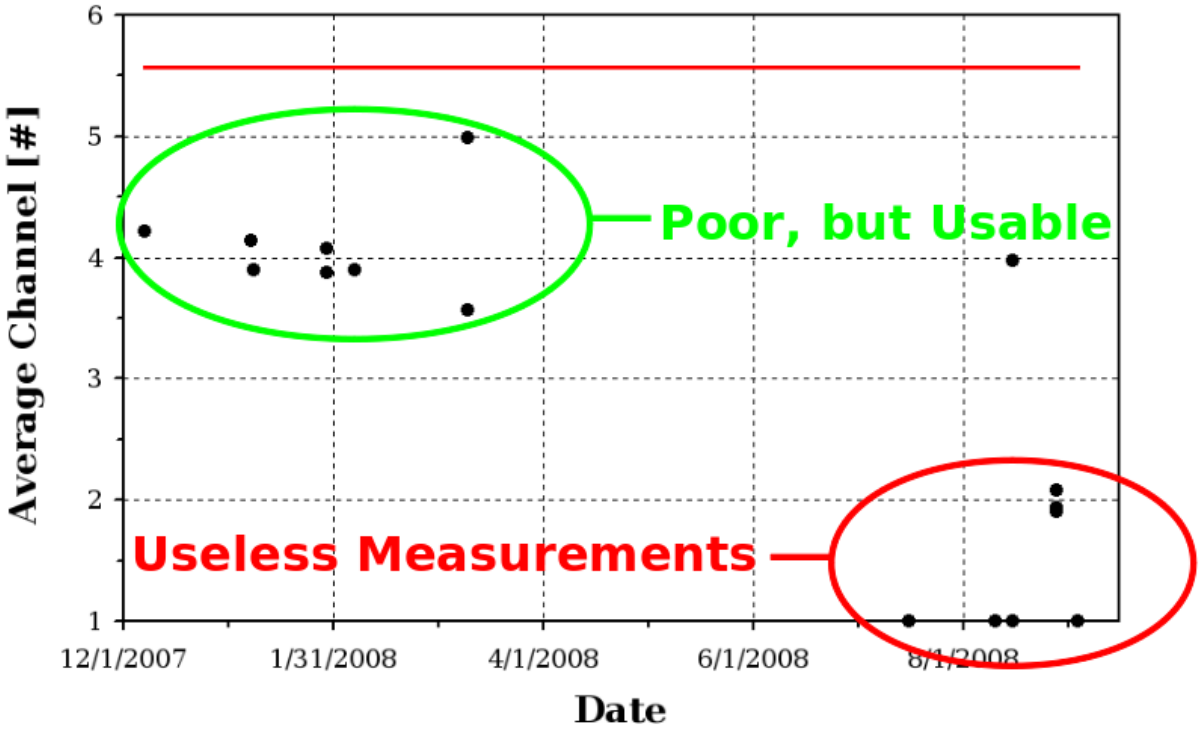

Fig. 3 Average channel values for the FSSP (Serial Number 1947-028-160) from performance checks conducted during the 2007/2008 field project. All calibration checks were performed in Saudi Arabia while the FSSP was on the research King Air 200 aircraft (N825ST). The solid horizontal line indicates the "standard" average channel value where 15 μm beads theoretically should be measured.

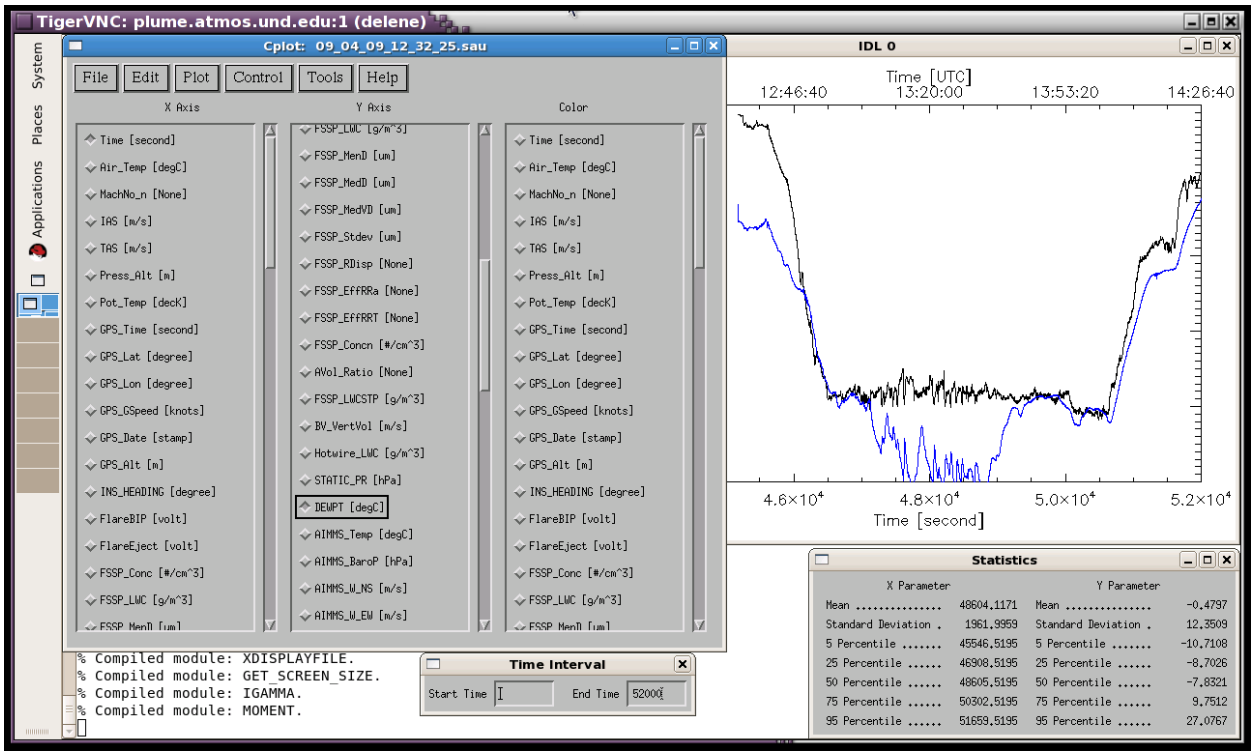

Fig. 4 Screen shot showing “Cplot” being used to analyze the 9 April 2009 summary data file.

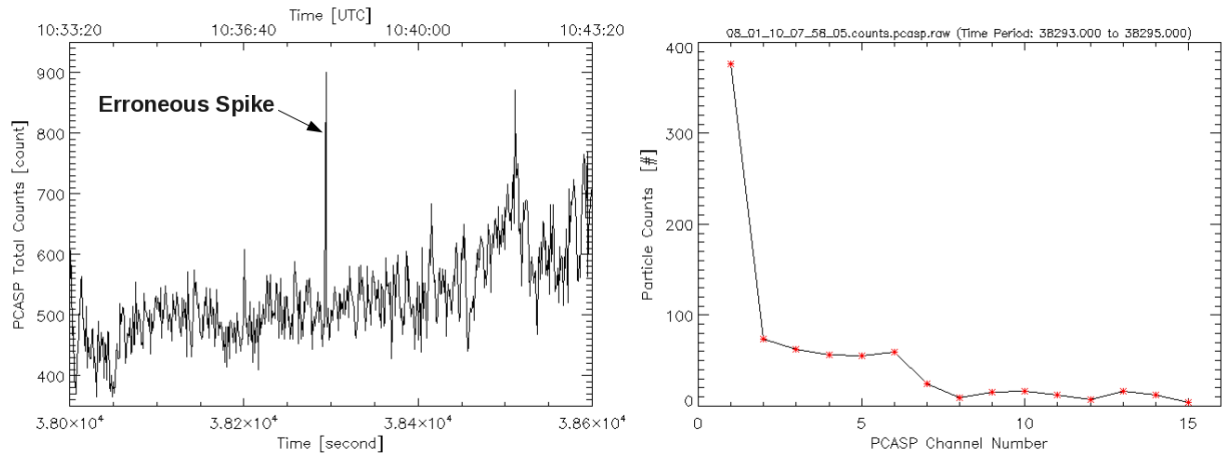

Fig. 5 PCASP total counts time series (left) and size spectrum (right) showing an erroneous spike during the first flight on 10 January 2008 in Saudi Arabia.

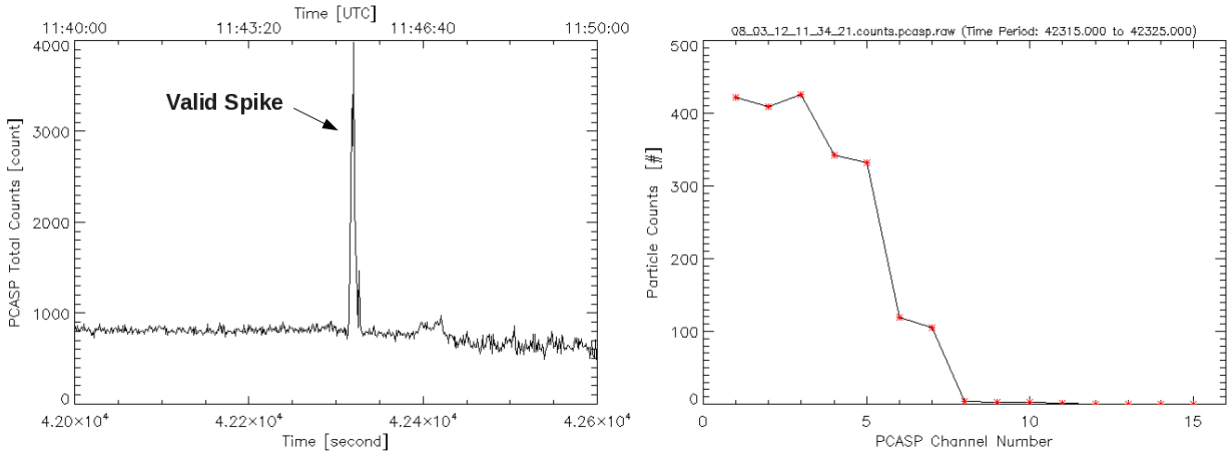

Fig. 6 PCASP total counts time series (left) and size spectrum (right) showing a valid spike during the 12 March 2008 flight in Saudi Arabia.

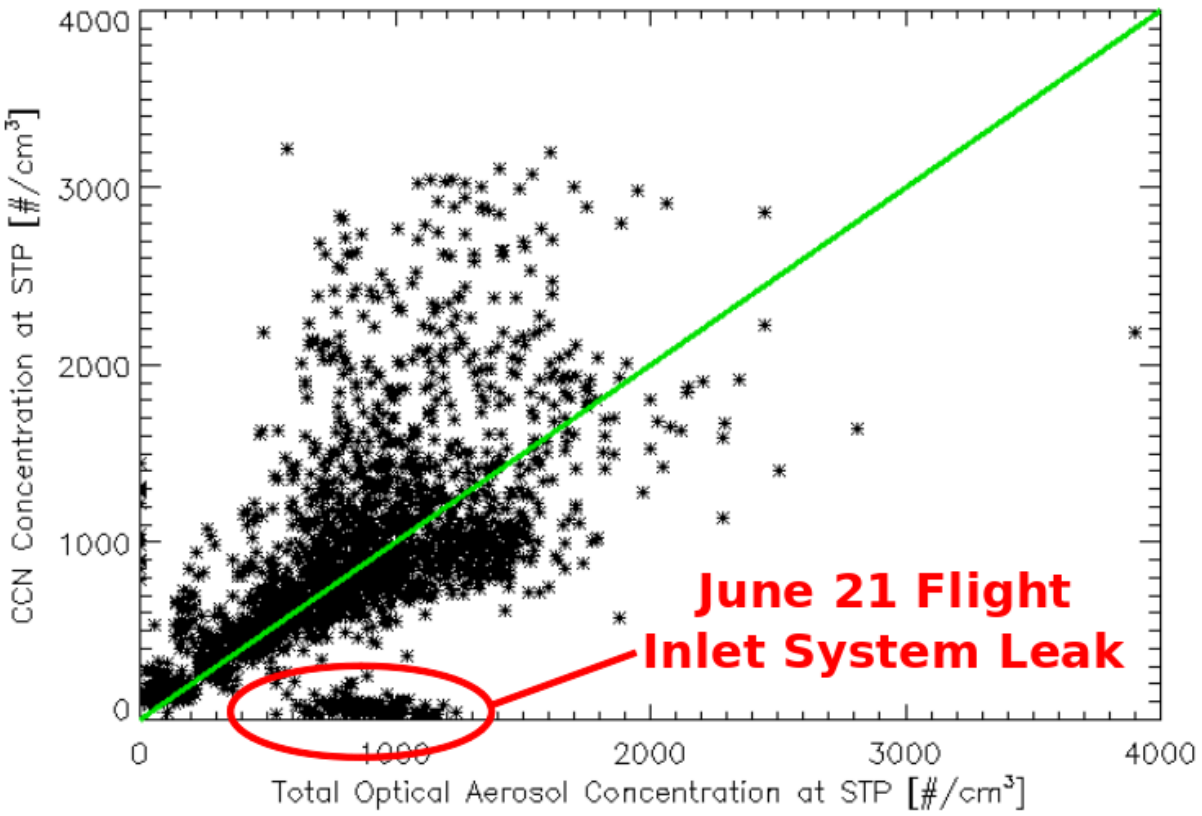

Fig. 7 The 1 Hz averaged total (0.1 – 3.0 µm in diameter) aerosol concentration measured by the Passive Cavity Aerosol Spectrometer Probe (PCASP) versus the University of Wyoming Cloud Condensation Nuclei (CCN) counter concentration (1% supersaturation). The solid green line is a one-to-one line. All valid out of cloud measurements (FSSP total number concentration less than 50 # cm^{-3}) obtained during the POLCAST2 field project are presented. Both the PCASP and CCN counter concentrations have been adjusted to standard temperature and pressure (STP).

Tables and Caption

Table 1 Comma delimited parameters that make up an ADPAA edit file.

Start Time	End Time	Short Name	Type	Year	Day Of Year	Author	Reason
Second from midnight when edit starts.	Second from midnight when edit ends.	Short name of the parameter to apply edit.	Type of edit to be applied.	Year the edit was made.	Day number of the year the edit was made.	Name of person making edit.	Why the edit is being made.

Table 2 Summary showing the time to compile ADPAA and the time to process a typical aircraft data set using ADPAA on a range of currently available computer systems. The processing time is for the Polarimetric Cloud Analysis and Seeding Test 2 (POLCAST2) field project which had 12 aircraft flights and 15 ground performance checks. The year given is when the computer hardware was purchased. All systems were running the Red Hat 5 Operating System (OS) with the exception of laptop, Convection, which was running the Fedora 12 OS.

Name (system)	Year	CPU	Cores	Memory	Compile Time	Processing Time
	yyyy	Type	#	MBytes	seconds	Minutes
Rayleigh (Workstation)	2005	Intel Pentium 4 CPU 3.00 GHz	1	894	44.6	209.90 (NFS Drive) 99.88 (Local Drive)
Buster (Workstation)	2006	AMD Athlon 64 CPU 3500 2.20 GHz	1	3,090	42.7	85.93 (NFS Drive) 77.70 (Local Drive)
Convection (Laptop)	2007	Intel Core2 Duo CPU T7700 2.40GHz	2	4,116	11.3	169.98 (USB Drive) 57.56 (Local Drive)
Plume (Server)	2008	Intel Xeon CPU E5205 1.86GHz	2	16,300	4.0	74.34 (NFS Drive) 68.96 (Local Drive)
Radar2 (Server)	2009	Intel Xeon CPU X5260 3.33GHz	2	4,116	2.7	41.28 (NFS Drive) 34.35 (Local Drive)