

MODIS SCIENCE DATA SUPPORT TEAM PRESENTATION

March 13, 1992

AGENDA

1. Action Items
2. MODIS Airborne Simulator (MAS) Status
3. Cloud Optical Program
4. Guidelines for Algorithms
5. Schedule

ACTION ITEMS:

08/30/91 [Lloyd Carpenter and Team]: Draft a schedule of work for the next 12 months. Include primary events and milestones, documents to be produced, software development, MAS support, etc. (Further modifications were made to the schedule.) STATUS: Open. Due date 09/27/91.

12/06/91 [Liam Gumley]: Investigate a cataloguing scheme for the MAS data. Consider the Master Catalogue, PLDS and PCDS. (Results of the investigation were included in the 02/14/92 handout. Review by the SDST was delayed.) STATUS: Open. Due date 02/14/92.

12/06/91 [Liam Gumley, Tom Goff, Ed Masuoka]: Develop a plan for storing and distributing MAS data. (The plan was included in the 02/14/92 handout. Review by the SDST was delayed.) STATUS: Open. Due date 02/14/92.

01/03/92 [Team]: Check on the set of software engineering tools available in Code 530 to see if any of these would be of use to the SDST. (Discussions were held previously with Frank McGarry of Code 530. Contacts were made with Joy Henegar, Stephanie Nickens and Linda Sheckler of Code 563.2 in regards to use of their PR:QA. The file dump algorithm and the cloud algorithm were processed using PR:QA. There were many warning messages which are being studied.) STATUS: Open. Due date 02/14/92.

01/17/92 [Tom Goff]: Have a polished version (with peer review) of the file dump routine ready for the MODIS Science Team Meeting. STATUS: Open. Due date 04/01/92.

02/21/92 [Ed Masuoka]: Talk to Code 930 and find out what tools they have for porting data between computers from different vendors. STATUS: Open. Due date 03/13/92.

02/21/92 [Lloyd Carpenter and Team]: Identify a list of risks associated with porting Team Members' algorithms to the PGS. Prepare these for discussion at the Science Team Meeting. STATUS: Open. Due date 04/01/92.

02/28/92 [Lloyd Carpenter]: Modify the SDST Schedule by adding a "Concept Development" activity and adjusting the start time of Team Member algorithm development. (The modified part of the schedule is included in the handout.) STATUS: Open. Due date 03/13/92.

02/28/92 [Liam Gumley]: Develop a plan to accelerate MAS processing using less of Liam's time. STATUS: Open. Due date 03/20/92.

MODIS Airborne Simulator status (Liam Gumley)

Progress up to 12 March 1992

(1) MAS data processing status

<u>Flight Date</u>	<u>Area covered during flight</u>	<u>Level-0 data received</u>	<u>Processing completed</u>	<u>INS offset fixed</u>
10/31/91	Ames test flight CA/NV	yes	3/3 tracks	yes
11/12/91	Ferry flight CA to TX	yes (subset)	1/1 tracks	no
11/14/91	Coffeyville KS	yes	16/16 tracks	no
11/18/91	Coffeyville KS	yes		
11/21/91	Coffeyville KS	yes		
11/22/91	Coffeyville KS	yes		
11/24/91	Gulf coast TX/LA	yes		
11/25/91	Coffeyville KS	yes		
11/26/91	Coffeyville KS	yes		
12/03/91	Gulf coast TX/LA	yes		
12/04/91	Gulf coast TX/LA	yes		
12/05/91	Coffeyville KS	yes	29/29 tracks	no
12/07/91	Coffeyville KS	yes		
11/16/91	Ground visible calibration	yes	10481 scanlines (no navigation)	
11/20/91	Ground visible calibration	yes	6078 scanlines (no navigation)	
11/23/91	Ground visible calibration	yes	10281 scanlines (no navigation)	

Tom Arnold indicated that the ground calibration datasets were of highest immediate interest, so these were processed to the Level-1B stage. It was agreed that the most useful form of the data would be to retain the Level-0 instrument counts in the Level-1B dataset, rather than calibrating to radiance units, since the data itself was to be used to generate calibration data. Thus it was decided to change the instrument configuration file CONFIG.ASC to indicate that ALL channels were visible, and thus should use pre-defined slopes and intercepts (as opposed to thermal channels calibrated from blackbody data). Since the radiances output in the Level-1B dataset are scaled to integers by

integer Level-1B radiance = nint(100.0 * real calibrated radiance)

the slope and intercept used for calibration were set to 0.01 and 0.0 respectively, so that the original instrument counts in each channel would be transferred to the Level-1B dataset. Also since no navigation data existed for these MAS datasets, the code was modified to allow processing to Level-1B without navigation data. Variables in the Level-1B dataset reserved for geolocation data were filled with the value -999.99.

(2) MAS Level-1B netCDF file dump utility

A utility has been written that allows a brief summary of a MAS Level-1B flight track file to be printed. The items printed were chosen from the list of items suggested as catalog entries for the MAS flight track files. The code is written in portable FORTRAN-77 and demonstrates how the netCDF library routines may be accessed through the FORTRAN interface. The code has been compiled and tested on both Silicon Graphics Iris and DEC VAX/VMS platforms, and has been included in the utility area of the MAS anonymous FTP

site. The following output shows the items printed for flight track 14 of the 05 December 1991 MAS (FIRE) flight. The system was the Silicon Graphics Iris.

```
% ../util/masdump
Enter MAS Level-1B netCDF file name : 05dec91-14.cdf
```

MAS Level-1B flight line summary

```
-----
Date                05-DEC-1991
Start time          163853.00 hours
End time            172241.00 hours
Nominal heading     319 degrees
Nominal altitude    19809 meters
Number of scan lines 16326
Start scan line number 49760
End scan line number 66130
Nominal BB1 temperature -38.17 C
Nominal BB2 temperature -0.31 C
Nadir start lat,lon 29.282, -93.748 degrees
Nadir end lat,lon 32.863, -97.008 degrees
Top left lat,lon 29.388, -93.602 degrees
Bottom right lat,lon 32.753, -97.156 degrees
Top left solar zen,azm 55.837, 155.618 degrees
Bottom right solar zen,azm 56.817, 164.181 degrees
```

```
%
```

(3) MAS image noise characterization/removal

Some preliminary work was done to investigate the nature of the coherent noise observed in MAS image data. Instrument examination and tests at Ames have indicated that the noise source is the MAS pod 400 Hz power and heater blowers (personal communication from Ken Brown, GSFC 925). Imagery from the 31 October 1991 MAS test flight was analyzed to check if this 400 Hz noise signal was present.

The first region examined was over the ocean off the CA coast. A FFT was performed on one scanline of the image data from channel 9 (4.5 micron) and the results are plotted overleaf. It can be seen that a sharp amplitude peak exists at around 400 Hz, with smaller peaks evident at approximately 1200 and 2800 Hz.

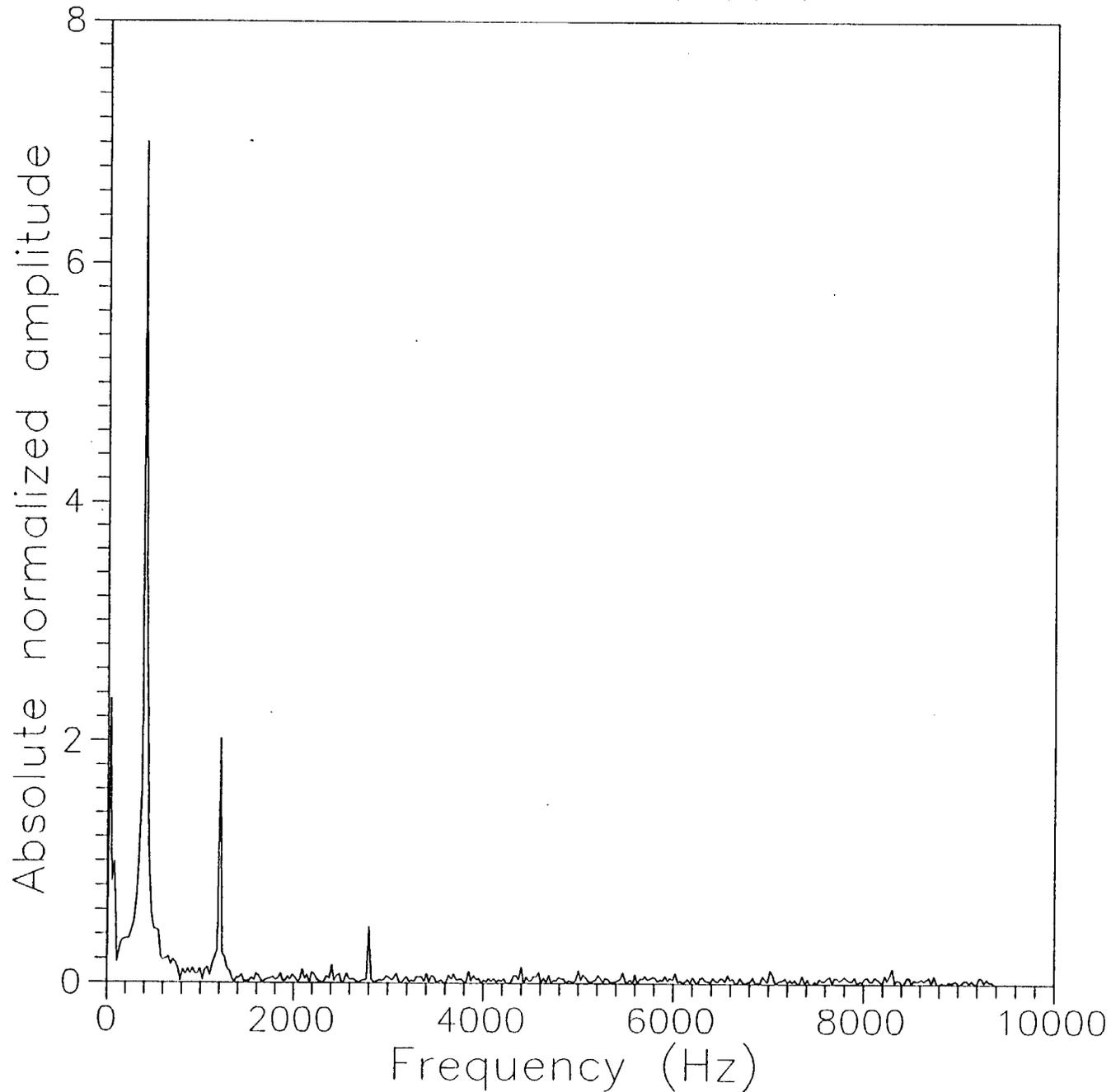
The second region examined was over land in CA (mountains, no snow). This spectrum (plotted overleaf) shows a higher information content in the image, but the 400 Hz signal is still dominant.

It should be recognized that the 31 October 1991 flight was for the purposes of engineering checkout, and Ames has since made modifications to the MAS to remedy these problems. However some noise is still evident in FIRE science flight data, and a means of removing that noise must be determined. At present, it appears that a two dimensional FFT approach would be most useful. This approach is being examined.

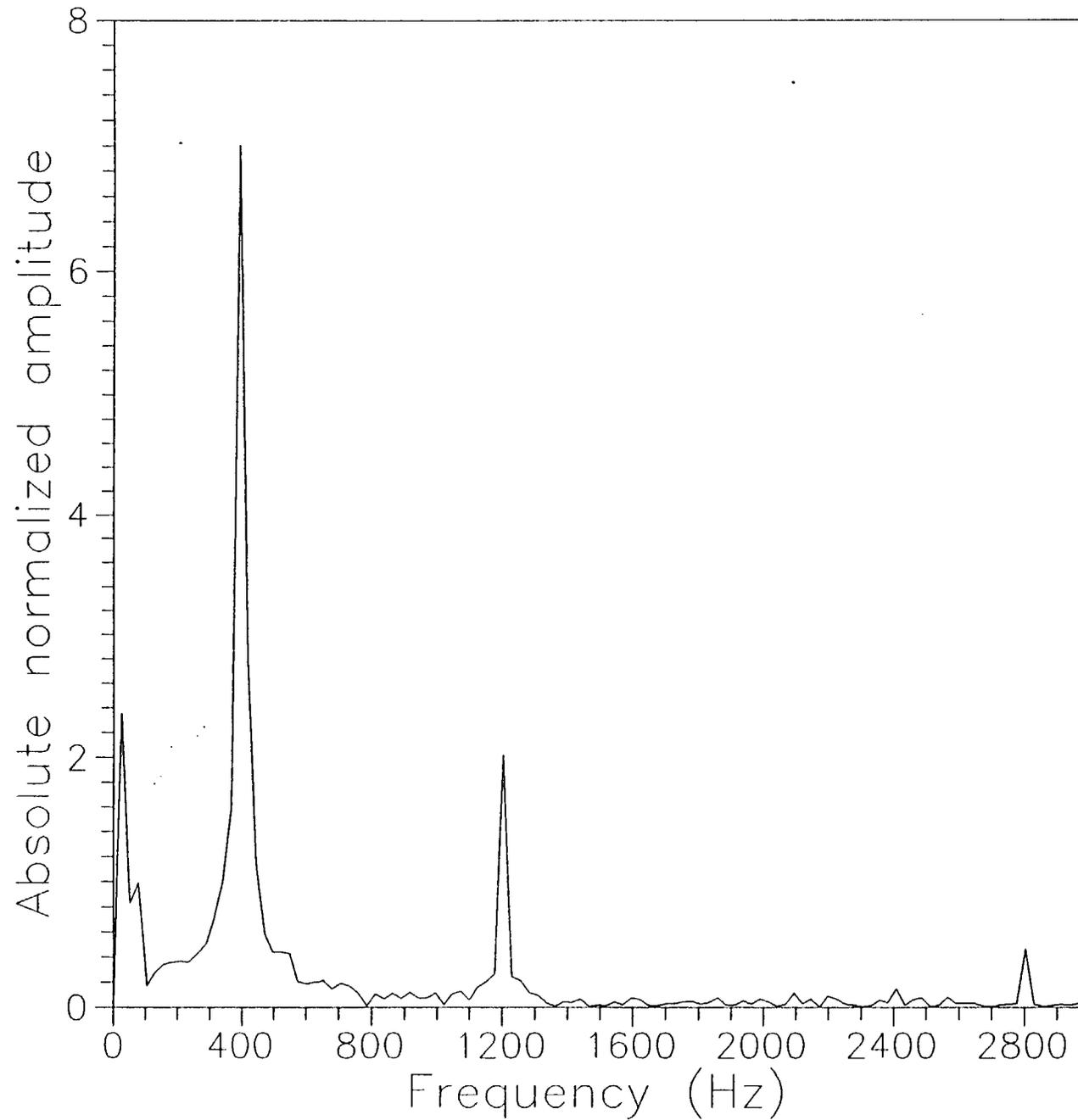
(4) MAS data distribution to the ASTER team

A phone inquiry was received from Simon Hook (JPL ASTER Team) regarding the availability of MAS data. It was explained that the data was currently available via FTP from GSFC. After obtaining Mike King's go-ahead, an introductory document describing the MAS data and instructions for retrieving it were emailed to Simon Hook (SIMON@PLDSJ1.JPL.NASA.GOV).

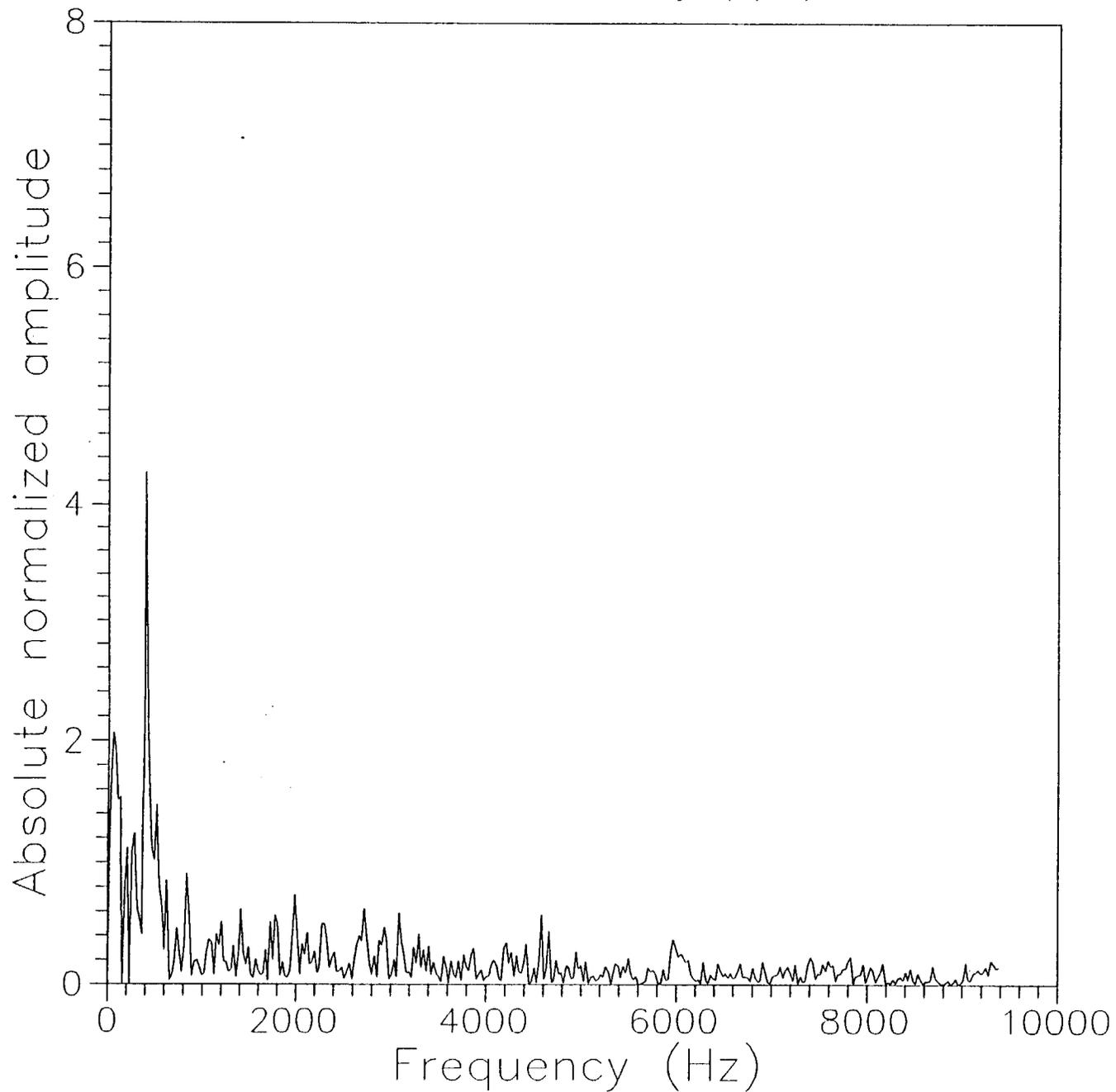
Fourier spectrum of MAS 4.50 micron scanline
31 October 1991 test flight over ocean
Frequencies $f(j)$ from $j=0$ to $j=(n/2)+1$, $n=716$



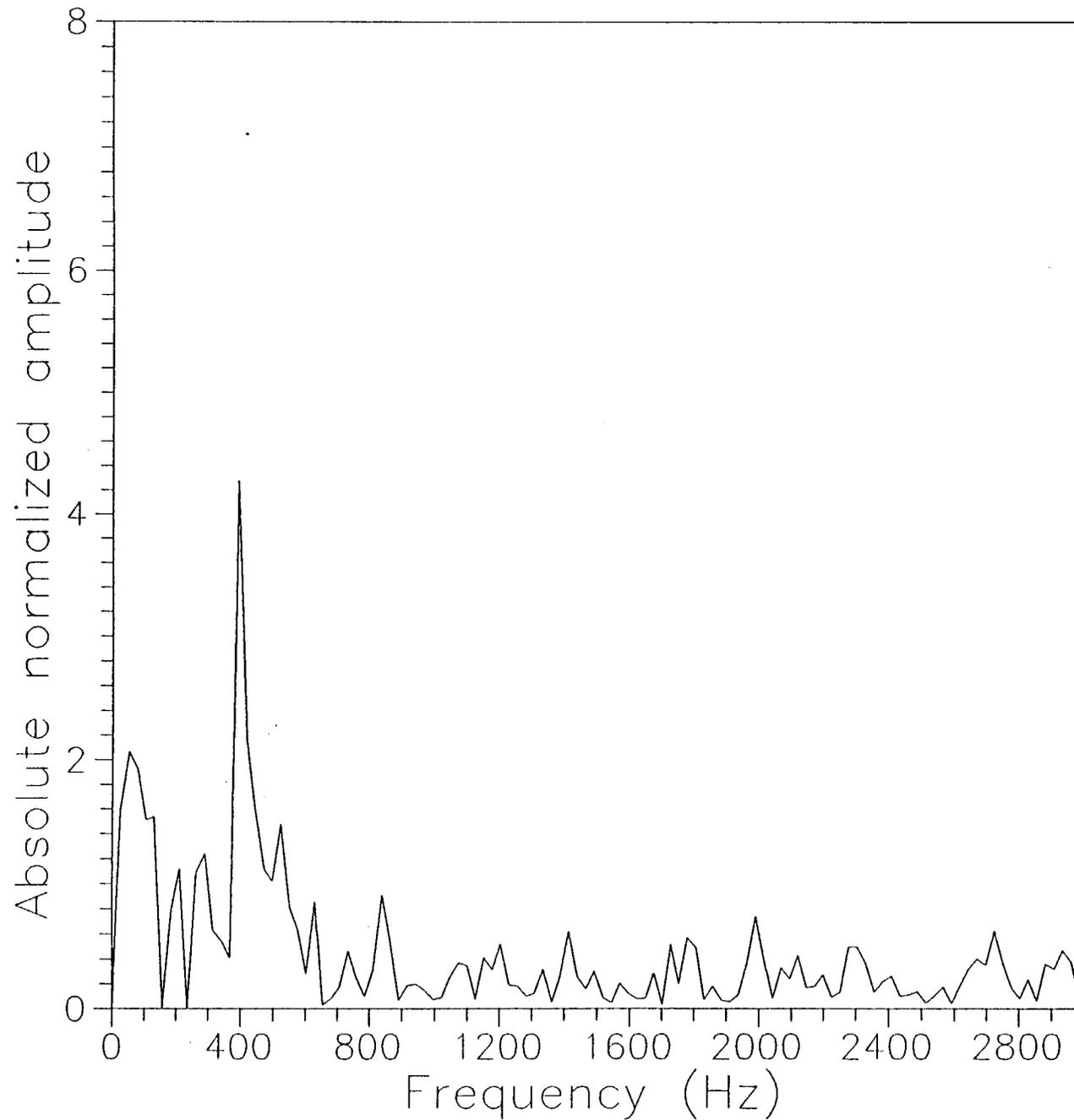
Fourier spectrum of MAS 4. J micron scanline
31 October 1991 test flight over ocean
Frequencies $f(j)$ from $j=0$ to $j=115$, $n=716$



Fourier spectrum of MAS 4.50 micron scanline
31 October 1991 test flight over land
Frequencies $f(j)$ from $j=0$ to $j=(n/2)+1$, $n=716$



Fourier spectrum of MAS 4.50 micron scanline
31 October 1991 test flight over land
Frequencies $f(j)$ from $j=0$ to $j=115$, $n=716$



Cloud Optical Program and MAS
Thomas E. Goff
12 March, 1992

tgoff on GSFC mail,
teg@ltpiris2.gsfc.nasa.gov,
or (301) 982-3704

- * **Cloud Optical Program Port, Continued** - Mike King's Cloud Optical Program (cldopt.f) is now executing on the SGI ltp iris computer. The port of the execution phase of this program was accomplished by modifying the code to place the large data arrays that had been placed into subroutine argument lists, into a common area. This was necessary because the SGI iris passes subroutine arguments by value instead of address and the large size of these arrays was producing a stack overflow condition. In addition, all necessary variables were explicitly set to zero upon program initiation. A related problem to uninitialized variables is the use of volatile subroutine variables by default. If a variable that is declared in a subroutine and used upon multiple entries to that subroutine, then there is no guarantee that the value of the variable is retained upon subsequent calls. Variables must be declared as type static to prevent their values from being changed when the subroutine is swapped to disk and brought back into a different part of physical memory before being called.

The original data set was designed to allow the program to terminate upon reading an EOF from the input data stream. The subroutine stubs that I wrote to interface the FTIO routines into the UNIX disk structure detected the EOF condition but could not pass this back to the FORTRAN calling routine via the FORTRAN error return extensions. The cldopt.f code was modified to bypass this extension. The original program further confused its internal variables which resulted in a termination due to a user specified IBM run time limit instead of the normal programmatical termination. This was corrected in the input data set to specify only one run case instead of the original two plus runs. The IBM 3081 execution was terminated by the operating system after 54 minutes of execution and produced two output cases. The SGI iris executed one case in 3 minutes. This equates to an approximate order of magnitude decrease in execution time on the iris when compared to the IBM 3081. The output data set that is sent to unit 6 was compared for the IBM and SGI iris runs and is identical.

This porting of the actual code took less than three days. However, the port of the data sets took approximately three weeks.

- * **MAS INS Data Plotting** - I have a hacked program on the SGI iris that will read the MAS *.ins data files and produce an output file that can be used as data to the PC based acrospin program. This will produce a real-time display of the 3-D aircraft flight path for any of the MAS flights that can be displayed, rotated, scaled, etc on any PC under user real-time control. Although this plot can not be used for determining the straight line flight paths, it

can be used to visually determine the coincidence of the race track segments and any elevation changes as the fuel load lightens. Ground coverage can be deduced from this display plot. If access to the Apple Mackintosh MacSpin program can be made available, I will be glad to provide a clone of the previous hack to produce MacSpin specific MAS flight track data sets to those parties who have an interest.

- * **MAS Graphic Overlays** - The MAS data has been ingested by the Khoros public domain imaging program which has been installed on Virginia Kalb's Sun 4. The data was ingested via my subset program, available on our anonymous ftp site. This program was then modified by Virginia to include a representation of the auxiliary data (lat-long, az-el, etc) as a graphic overlay to the original data. When a better facility for obtaining hard copy of these images is available, we will be able to produce quality assurance products to validate the ground location and other ancillary data. Note that Khoros will only run in a computer that has the X11 rev 4 windowing facility. This capability will be expanded, on a time available basis, to include software tools slanted toward the remote sensing disciplines.

-- Miscellaneous --

- & **VAX Cluster Modem Lines** - 53241 now appears to work consistently.
- & **UNIX Mail** - The SGI mail facility succeeded in transferring mail to/from an outside computer on the internet.
- & **SLIP Connection** - I have revision 3.5 of SUN's PC-NFS installed on my PC. I will now establish the procedures required to connect to the GSFC NCCS slip server. This is a non-standard implementation due to the existence of the Rohm switching system and may take some time and patience to make fully functional.
- & **FORTRAN Source Code Checking** - I have a trial evaluation copy of FORTRAN-LINT arriving by UPS which I would like to install on the SGI ltpiris2 computer. This software requires the X11 rev 4 windowing facility and can be installed when this facility is available.
- & **Current Source Code Checking Facilities** - We plan to be using the LINT and public domain FTNCHEK programs to check our C and FORTRAN code for non-portable or wasteful features. The LINT program on the iris complains of an undefined variable which is defined by the operating system in the <stdio.h> include file. This needs to be exorcised by SGI in the future.
- & It would be very useful, especially for the off-site people, to have a synopsis of the ltp computer facility. This could include, but not be limited to, the following:

A list of the computers with the operating system type, the current revision level of the operating system, and if a UNIX based system, which UNIX (BSD or AT&T) was used as the basis for the vendor supplied UNIX clone. Also which windowing or GUI is provided on the computers.

The physical location of the computers and what means of accessibility they have

(locked rooms, etc).

A list of available peripherals with locations and how to access these devices ala help files etc. This would include the location of post script, HP pcl, and HPGL devices; and magnetic tape systems with density, format, and network or direct computer access. How do we obtain image B/W or color hard copies, and what software is available for image format conversions?

A list of major software that is available on the various ltp computers. We have interest in publishing tools, word processors and word processing translator capabilities, image processing packages with a list of capabilities (classification techniques, panable graphics overlays, user specifiable translation equations, etc), data representation tools (2 and 3-D plotting, contour plotting, FFT's and similar transforms), spreadsheet and similar columnar data translating programs and editors, library tool kits for math and general data manipulation, etc.

A list of direct contacts for obtaining additional information on networking, operating system and computer architectural problems, software usage, ftp sites, usenet connections for public domain, shareware, and freeware programs, computer system factory support (which systems are supported, by whom, with what level of response), etc.

c:\modis\status.wp

DRAFT

**MODIS Science Data Support Team (SDST)
Guidelines for MODIS Team Members Science Algorithms**

DRAFT

Table of Contents

1	Introduction	1
2	Readability	2
3	Portability	2
4	Modularity/Cohesiveness	3
5	Testability	3
6	Maintainability	4
7	Documentation	5
8	Usage of Software Tools	5
	Appendix A: Software Code Evaluation Criteria	7
	References:	9

DRAFT

1 Introduction

The objective of these guidelines is to facilitate the porting, integration, testing, documentation, and maintenance of algorithms for the generation of MODIS science data products on an operational basis. The MODIS Science Data Support Team (SDST) provides support for the development of the MODIS science data processing system. The SDST will generate a plan for the development, validation, integration, operational testing, documentation, maintenance, modification, and configuration management of the MODIS science data processing algorithms. These guidelines are designed to assist the MODIS Science Team Members in preparing their algorithms for this process. The evaluation criteria given in Appendix A are intended to provide a check list for examining code to see if it is reasonably consistent with good coding practices.

All MODIS science data processing algorithms to be ported to the MODIS Team Leader Computing Facility (TLCF) and integrated into the shell with other algorithms will be written in either the Standard Fortran or Standard C programming language, without extensions. Any exceptions should be brought to the attention of the MODIS Team Leader and coordinated with the SDST.

The EOS Product Generation System (PGS) and the MODIS TLCF are planned to run under the UNIX operating system. For MODIS science algorithms which are developed and tested under a different operating system, it is especially important that they be written in terms of a small set of primitive operations for accessing the environment. If the primitives can be implemented on a UNIX system, then it should be possible to port the algorithms to a UNIX system without great difficulty. This process is especially important for algorithms developed under an operating system other than UNIX.

The primary goal of these guidelines is to facilitate the porting, integration, testing, documentation, and maintenance of algorithms for the generation of MODIS science data products on an operational basis. These guidelines must be consistent with accepted good coding practices, but they are not intended to go beyond the present purpose. Choice of style, computational methods, etc. are left to the Science Team Members.

DRAFT

2 Readability

Efficient porting, integration, testing, documentation, and maintenance of algorithms depends heavily on being able to read and understand the code (including the documentation). Ready access to the originator is helpful but not sufficient. The code must be written and documented so as to be read, understood, and used by another person or group of people in a different environment. Readability, or understandability, is a very important criterion for successful implementation of MODIS algorithms. Readability is dependent upon:

- good documentation
- meaningful naming of variables
- clear structure and logical flow, and
- appropriate modularization.

3 Portability

Portability is of primary concern in the case of MODIS science algorithms which are developed in one environment and implemented for production processing in a different environment. The degree of concern increases with the degree of difference between the two environments.

The best guideline is to adhere to standard FORTRAN or C, and resist the temptation of system-dependent features.

Many portability problems are best avoided by specifying a small set of primitive operations for accessing the environment. Operating system dependencies are then confined to a small number of procedures and functions, so the algorithm can be moved to any system where the primitives can be implemented.

Special care must be taken to avoid problems associated with differences in binary data files. Another concern is the change in roundoff arising from differences in the bit-length of floating point arithmetic in different computers.

DRAFT

4 Modularity/Cohesiveness

Complex algorithms can generally be separated in a logical way into modules (subprograms) each of which does a single task. When each module is cohesive, and the coupling between modules is loose, the algorithm becomes more understandable, more testable, more maintainable, and more easily documented. The aim is to achieve a level of modularity which avoids the clutter of too many relatively trivial modules, while keeping the individual modules cohesive and comprehensible. Avoid combining several functions together arbitrarily. Modules should generally be limited in length to 1 or 2 pages.

5 Testability

The porting and integration process is not complete until the algorithm has been successfully tested on the TLMF. These tests should be consistent with testing done prior to delivery to the SDST. All test data and test results must be included with the delivery of the algorithm and documentation. The testing process should include:

- testing all modules independently of the algorithm itself:
 - for reasonableness of results
 - against limiting cases with analytic solutions
 - against published results where possible
 - exercising every logical branch
 - for possible failure
- testing for invalid and implausible input variables in all modules, no matter how unlikely it is that the module will be used incorrectly, and
- provision of test drivers
 - Each algorithm should be distributed with comprehensive test drivers that explore all of the major branches of the algorithm.

DRAFT

- .. Test drivers should be developed with the same good coding practices as the algorithm itself.
- .. Test drivers should "failure-test" the model by pushing it into regimes where trouble is expected.
- .. "Correct answers" should be included in the driver code.
- .. A minimalist approach should be taken. There is no need to generate large print-outs.
- .. Each test case in the driver should be completely independent of the others.
- .. Test drivers should be critically evaluated in the same way as the algorithms.

6 Maintainability

Considering the duration of the MODIS mission, all algorithms will be subject to maintenance (changes, and updating). The maintenance process will be greatly simplified by using forethought in the design and development of the algorithms. Most of the guidelines addressed elsewhere in this document will improve the maintainability of algorithms. Some additional considerations are:

- "Localize" the algorithm so that the range of influence of any change is small.
 - .. Keep variables localized by assigning them (giving them values) close to the place where they are used.
 - .. Keep the logic localized.
 - .. Within reason, use short modules that communicate only through argument lists. Minimize the use of global variables, COMMON blocks and EQUIVALENCE.
- Strive for maximum generality so that fewer changes are required.
 - .. Consider dimensioning all arrays with symbolic constants.

DRAFT

- .. Use logical or integer flags in the argument sequence to select among various options.
- .. Concentrate system-dependent file manipulation, etc. in a few well-chosen general modules that call low-level primitives to do the work.

7 Documentation

Comments are interspersed throughout a module. Documentation, on the other hand, occurs only at the beginning of a module. It is specific, and it should be revised to reflect changes.

Completeness, not brevity, is the main consideration. The documentation should, at a minimum, consist of:

- purpose of the module
- definition and units of the input and output variables
- description and units of the important internal variables
- method used, if applicable
- references to the literature
- notes and warnings (conscious design limitations).

Thorough and complete input variable definitions and descriptions are especially important. Units, type, dimensions, special cases, upper and lower limits, usage examples, default values, and interrelations with other input variables should all be provided.

8 Usage of Software Tools

Software tools are modules of modest size that:

- solve a general problem, not a special case;
- are nearly perfect, having gone through a considerable shakedown process before being released; and
- are user-friendly enough that programmers will prefer them

DRAFT

to building their own.

The idea is to create more complex programs mainly by combining tools in different ways. Tools can be divided into categories as utility, numerical analysis, or scientific.

DRAFT

Appendix A: Software Code Evaluation Criteria

This list of software code evaluation criteria is provided as a suggested checklist for code review.

- Does each module have the standard prologue?
- Does each module contain a single entry point and a single exit point?
- Are declarations and data statements placed after the prologue and before the first executable statement?
- Are literal constants not embedded in executable statements?
- Do arguments in call statements not contain arithmetic or logical expressions?
- Has structured programming been utilized?
- Are limit checks performed to ensure that variable contents are within the expected range of values?
- Are input defaults explicitly tested?
- Are unique values of keywords checked?
- Are constants defined? Are counters and parameters initialized?
- Are loop index parameters and array subscripts expressed only as integer constants or integer variables?
- Are variable names meaningful?
- Do statement labels begin with a specified value and occur in an obvious sequence?
- Is the code designed to handle errors gracefully?
- Are shared variables communicated as arguments whenever practical to ensure program modularity?
- Is code adequately commented and indented to show structure?

DRAFT

- Are data sets properly opened and closed?
- Are flagged/missing data values correctly excluded from calculations?
- Is there a path defined for every possible outcome of a logical decision?
- Are leading and trailing records recognized and handled appropriately?
- Are correct units or the appropriate conversion used?
- Are the correct signs (positive or negative) used?
- Are display formats large enough?
- Are frequently utilized values stored after they are calculated?
- Are double-precision constants used in double-precision expressions?
- Is specified precision sufficient for required accuracy?
- Is the level of nesting kept to a minimum?
- Are possibilities for infinite loops avoided?
- Are errors and associated error messages kept together?
- Are functions and closed subroutines used only when they are more efficient than in-line code?
- Does the coded module satisfy the standard characteristics of the implementation language, for example:
 - Is code logically blocked and indented?
 - Is there only one statement per line?
 - Do variable names avoid the use of language keywords?
 - Are comments uniformly set-off from code?
 - Are all transfers of control and destinations annotated?

DRAFT

References:

Warren J. Wiscombe, "Principles of Numerical Modeling with Examples from Atmospheric Radiation" 1989, unpublished.

Upper Atmosphere Research Satellite Software Assurance Recommendations Document, Version 3.0, NASA/GSFC, July 1989

