PARALLELIZATION OF ROCKET ENGINE SIMULATOR SOFTWARE

(P.R.E.S.S.)

/IN-61

[illegible]

RESEARCH SUMMARY REPORT

Principal Investigator

Dr. Ruknet Cezzar, Associate Professor
Department of Computer Science
Hampton University
Hampton, Virginia 23668
email: cezzar@cs.hamptonu.edu

Technical Officer

Dr. Don Noga
Mail Stop 501-2
NASA Lewis Research Center
21000 Brook Park Road
Cleveland, Ohio 44135

Grant Number: NAG3-1792

Grant Period: 3 Years

Start date: October 19, 1995
End Date: October 18, 1998
Extention Date: October 18, 1999

(October 18, 1998)

TABLE OF CONTENTS

1. Background

Parallelization of Rocket Engine System Software (PRESS) project is part of a collaborative effort with Southern University at Baton Rouge (SUBR), University of West Florida (UWF), and Jackson State University (JSU).

The third-year funding, which supports two graduate students enrolled in our new Master's program in Computer Science at Hampton University and the principal investigator, have been obtained for the period from October 19, 1997 through October 18, 1998. The interim progress report dated April 21, 1998 outlines the plans and progress in its relevant sections:

Background

Research Progress Overview

Current Research Activities (October 1997 through April 1998)

Future Plans

Conclusion

A noteworthy activity since that time was no-cost extension for an additional year with revised budget. The revised budget provides slightly more attractive stipend and tuition support for two graduate research assistants and provides release time and full time summer support for the principal investigator. Although, as shown on the authorization letter (See Appendix A) on July 7, 1998, the internal budget by Hampton University's research and development office has only recently been entered into the computer system. This is due to the revised budget applying only to FY 99, starting on October 19, 1998 for this project. The project has fallen behind in spending rate (See Appendix B). We discuss this in more detail in Section 4.2.

The key part of the interim report dealt with the acquisition of the working version of GASP (gas properties) module, referred to as GASPSheer. That was necessary for debugging and obtaining an executable for PUPMDES/TURBDES software which has been lent earlier to Hampton University under a software agreement. There is also the discussion of PI's presentation at the Fifth Annual HBCU Conference, April 9-10, 1997, Cleveland, Ohio. The presentation basically outlines distributed computing strategies for NASA's Fortran-based software, in particular, the RENS and NPSS software at Lewis Research Center. The presentation focuses on the major challenge involving lack of support for Fortran-based software by distributed computing standards and software tools such as CORBA (Common Object Request Broker Architecture). As its name indicates, this most popular distributing computing standard, only supports object-oriented languages such as C++, Smalltalk, and Java. More recently, it also provides limited support for Ada. Finally, there was some discussion of the work on C++ wrappers, in our case, using Microsoft Fortran 5.1 and Borland C++ 5.0, on PC platform.

2. Research Progress Overview

In this section, we give an overview of progress since the interim report filed in April 1998. As mentioned in the interim report, an important aim for this period was experimentation with MPI (Message Passing Interface) toward a demonstration of distributed computing over a local area network. Initially, the aim was to use a working version of PUMPDES or TURBDES software using the new GASP obtained from NASA LeRC with a new software agreement (See Appendix C). Later, with acquisition of what became known as SOURCCDS-folder modules under another software agreement (See Appendix D), it became clear that using the main executable module of that package would be more productive. That main module is RESSAP.FOR, where RESSAP is an acronym for Rocket Engine System Software Analysis and Performance. RESSAP.FOR is a preliminary source code, and has not been thoroughly tested. However, RESSAP calls both PUMPDES and TURBDES modules, as well as other modules. Thus, it appeared that RESSAP would be a good candidate for experimentation with MPI.

2.1 Message Passing Interface (MPI) Refresher

MPI and its implementation on Hampton University's SunOS based LAN have been discussed in detail in the Research Summary Report submitted on September 2, 1997. As a recap, we should briefly mention that it is a message passing interface standard intended primarily for tightly-coupled distributed-memory systems. However, a SunSparc version MPICH exists, and this implementation provides a standard protocol, through a library and executable shells, for message exchanges. Note carefully that the underlying issues of remote execution is to be done by the system, somehow, some way. Through Unix shell scripts (e.g. MPIRUN), and using SunOS and Sun's NFS (Network File System) facilities, it is possible to assign processes to different network nodes. Indeed a cursory demo package which can also run on ACCL LACE cluster have been discussed in detail at that report. There, sample demos for MPI, as well as PVM, were discussed. The computation involves a two-dimensional 50x50 grid using Jacobi's algorithm to solve the Laplace's equation. The ultimate aim of that computation is to solve a linear partial differential equation.

The best source for MPI is shown in [3]. There some 128 routines and how those standard library routines can be invoked are discussed. The parameter blocks of calls such as MPI_SEND, MPI_RECV, MPI_SENDRECV_REPLACE, etc. are long. They are also highly complex since the arguments are IN, OUT, or INOUT. At any rate, the focus of this reference source is on the intricacies of message passing among process groups and among processes within process groups. The discussion pays no attention whatsoever on the implementation issues involving various hardware problems. In effect, all the examples involve a single multiprogrammed processor node which run processes concurrently. Various fine issues such as deadlocks, barrier synchronization, producer-consumer problem and the like are discussed in great detail. Additionally, there are various calls for error handling and profiling. The 128 routines which unfortunately makes MPI huge and complicated also involve various Fortran 90 type constructs for data-parallel execution of loops involving arrays. In a sense, the focus is on fine-grain parallelism which is also supported by Fortran 90. For instance, one of the best examples, and there are only few examples, involve Jacoby iteration (Section 2.5, pp. 39-43).

Although MPI is intended for fine-grain parallelism with symmetric topologies of process groups, such as two or three dimensional meshes, we intend to use it for course grain parallelism. The main idea is to have an executable Fortran code, such as RESSAP (Rocket Engine System Sizing and Analysis Program), and use MPI to run its procedures on different network nodes. In addition, as will be discussed shortly, the fine grain parallelism in some of the modules, such as TABLE and NEWTON, can be exploited via so called intracommunicators. However, there is a question mark on that issue, since message passing delays are significant in a LAN. Therefore, the fine-grain parallelism over a LAN will probably incur too high a communication overhead. The suitability of MPI for parallelization of GASP which is called most frequently by PUMPDES/TURBDES software, is unclear to us at this time.

2.2 Analysis of the RESSAP Code

Our familiarity with this code came from the content of SOURCCDS which, as mentioned earlier, was lent to Hampton University. In that directory, there are various source code modules left over from Dean Scheer's work. Through experimentation, and using the code from the earlier TURBDES and PUMPDES system, we were able to obtain an executable a.out which corresponds to RESSAP main module. The listing in Figure 1 gives all the source modules along with inputs and outputs. We must mention that there is no documentation about RESSAP such as hierarchy charts showing the logical relationships and interfaces.

INPUT	MODULES	OUTPUT
CPGG.DAT	ressap.for	for015.dat
CPLL.DAT	pumppd.for	fort.20
CPTP.DAT	turbpd.for	
CSTARGG.DAT	gasp.for	
CSTARLL.DAT	fanno.for	
CSTARTP.DAT	setup.for	
DEFALT.DAT	sgasp.for	
DFTPMP.DAT	xbldlg.for	
DFTTRB.DAT	xcstar.for	
GAMMAGG.DAT	xenth.for	
GAMMALL.DAT	xispid.for	
GAMMATP.DAT	xleak.for	
INTURB.DAT	xnewton.for	
ISPTHPERGG.DAT	xpfit.for	
ISPTHPERLL.DAT	xpmphd.for	
ISPTHPERTP.DAT	xreadp.for	
MACHGG.DAT	xstagt.for	
MACHLL.DAT	xtable.for	
MACHTP.DAT	xtcapd.for	
MACP.DAT		
MK480TRB.DAT		
PRGG.DAT		
PRLL.DAT		
PRTP.DAT		
REINPT.DAT		
RL10STG2.DAT		
TSTAGGG.DAT		
TSTAGLL.DAT		
TSTAGTP.DAT		
VISCGG.DAT		
VISCLL.DAT		
VISCTP.DAT		

Figure 1: Source modules for a working version of RESSAP software

In this configuration, PUMPPD.FOR and TURBPD.FOR correspond to the previous versions of PUMPDES and TURBDES software. Note that, even though there is a calling sequence, as the debugging trace shown in Appendix E shows, these two modules are largely independent of one another. However, they do call and share various other subroutines as shown in Figure 2.

```
PUMPPD.FOR:              TURBPD.FOR:
-------------            -------------

CALL SETUP               CALL SETUP
CALL GASP                CALL GASP
CALL GASP                CALL SETUP
CALL SETUP               CALL GASP
CALL GASP                CALL SETUP
CALL GASP                CALL GASP
CALL GASP                CALL TABLE
CALL TABLE               CALL TABLE
CALL NEWTON              CALL GASP
CALL GASP                CALL GASP
CALL NEWTON              CALL TABLE
CALL GASP                CALL GASP
CALL LEAK                CALL GASP
CALL GASP                CALL TABLE  (appears 12 times in 12 places)
CALL GASP                CALL NEWTON
CALL BLDLG               CALL GASP
CALL PMPHD
CALL GASP
```

Figure 2: PUMPPD and TURBPD modules sharing other sub-modules.

As is clear, SETUP, GASP, NEWTON, and TABLE subroutines are shared. The SETUP is a plotting program for the old VAX/VMS system, and therefore, need not be considered. Instead of leaving it out, we stubbed it. Thus, a second tier process group involving just GASP, NEWTON, and TABLE can be established. To reiterate, the first process group consists of only 2 processes: PUMPPD and TURBPD. This is a highly unconventional use of MPI since, as mentioned earlier, MPI is most suitable for a very large number of processes organized symmetrically, such as the grid structure.

As the first step, we needed an executable which runs to completion and reports a very large amount of numerical data in for0015.dat. If there are error message, such as non-convergence reported by NEWTON subroutine, those are output on fort.20. Unfortunately, perhaps due to migration from VAX/VMS based system to Unix, there are arithmetical exceptions having to do with zero divides throughout the code. We have spent a great deal of time in debugging so as to be able to at least see what subroutines are run and in what sequence. Appendix E gives a sample of the kind of debugging trace we attempted by embedding Fortran PRINT commands in appropriate places. This also gives a better idea about how, for instance, PUMPPD and TURBPD run the second tier sub-modules, including NEWTON and TABLE which are shared.

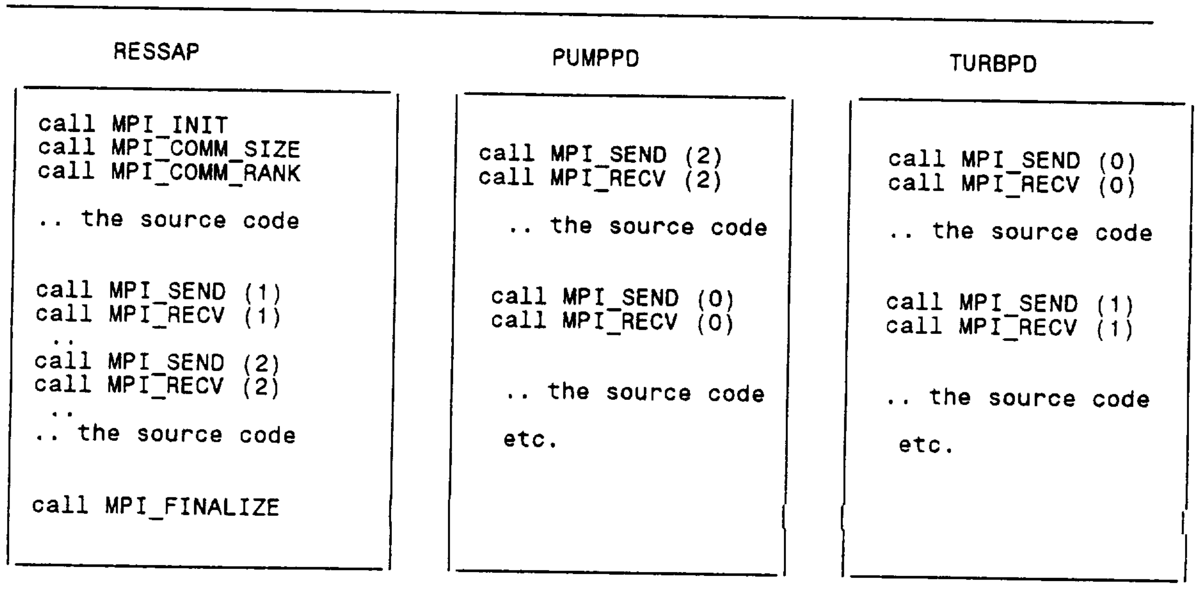
2.3 Using MPI for RESSAP Code

From the analysis above, it is best to use MPI's point-to-point communication interface in between the two modules PUMPPD and TURBPD. In this case, there will be MPI's intercommunication mechanism for the two sides of the point-to-point communication link. Having established message exchanges between PUMPD and TURBPD. One important parameter is MPI_COMM_WORD which is the default communicator that specifies the communication domain. This depends on the environment and specific implementation. For instance, suppose we have used the following process group:

File name: pgroup

```
apple 0 /users/cezzar/press/codes/ppp/ressap
lemon 1 /users/cezzar/press/codes/ppp/ressap
basil 1 /users/cezzar/press/codes/ppp/ressap
```

In this case, MPI_COMM_WORLD will be established as the host machine apple with id 0 initiating the run and the access to the other two machines through SunOS remote execution facility. Moreover, as a result of MPIRUN shell script, at least 3 initial processes, with ids 0, 1, 2, .. will be set up. As is mentioned earlier, the emphasis of MPI is with the intricacies of the logical aspects of process communications without attention to implementation issues. Once this is done, Figure 3 shows schematically and in an oversimplified way, how the MPI function calls may be embedded in the three modules. The numbers 0, 1, 2 refer to the process ids, where process 0 is the controlling process which must call MPI_INIT and only once.



RESSAP

```
call MPI_INIT
call MPI_COMM_SIZE
call MPI_COMM_RANK

.. the source code

call MPI_SEND (1)
call MPI_RECV (1)
 ..
call MPI_SEND (2)
call MPI_RECV (2)
 ..
.. the source code

call MPI_FINALIZE
```

PUMPPD

```
call MPI_SEND (2)
call MPI_RECV (2)

 .. the source code

 call MPI_SEND (0)
 call MPI_RECV (0)

 .. the source code

 etc.
```

TURBPD

```
call MPI_SEND (0)
call MPI_RECV (0)

.. the source code

call MPI_SEND (1)
call MPI_RECV (1)

.. the source code

 etc.
```

Figure 3: A Schematic depiction of MPI function calls in RESSAP

Naturally, the MPI calls have a highly complex parameter structure where each parameter is of type IN, OUT, and INOUT. In addition, the message is strongly typed, where it is difficult to pass arguments when Fortran subroutines are converted to MPI processes. The following discussion elaborates on this issue.

The embedded MPI call (See Appendix F for details) for sending a message to TURBPD and receiving an acknowledgement was, in effect, a substitute for the subroutine call:

```
CALL TURBPD(KFLUID,KTRBH,KINPT,NSTGH,P1HTRB,T1HTRB,HPHPMP,
1 WTRBH,RPMH2P,UTMH2T,AOVRDH,CNOHNH,ALPH2H,CLRH,HPTRBH,DMTRBH,
2 UMTRBH,UCTRBH,ETATSH,ETATTH,H1HTRB,D1HTRB,P2HTRB,T2HTRB,
3 H2HTRB,D2HTRB,S2HTRB,KTYPEH,ANSQDH,ADMH,Z1H,CP1H,GAM1H,
4 UTTRBH,HTR1H,HTR2H)
```

In the example, because MPI requires typed messages, we were able to send only the first argument KFLUID as an integer value. Moreover, we used the tag field of the message to indicate to the other process that this is the first argument. What if we wanted to pass all of the arguments to TURBPD process. Using a separate MPI_SEND call for each argument, with the appropriate tags indicating which argument, would be unwieldy. In the case of Fortran, it is not possible to pass the parameters as a block and sending just the pointer to that block. That is a dangerous practice even if allowed, as with C. The best solution, in the case of Fortran, is to pass all the parameters as a COMMON block. If for some reason that cannot be done, we have no choice but prepare an array for each sequence of arguments of the same type (e.g. say REAL*8) and then pass that array to the receiving process. Note that, in the case above, we cannot do this for all the arguments since the arguments are of different types.

Therefore, there may need to be a reordering of arguments, where say all integer arguments first, then all real arguments, and so on. In that case, for each sequence, as separate Fortran array needs to be prepared. Thus, if many arguments need to be passed to another process, the coding of MPI calls will be highly complex. We may be tempted to pass the list of names of these arguments, as shown above, as a string in a single message. Aside from limitations on message length, such an approach would accomplish little since what is required is the values of the arguments.

2.4 HBCU Conference Attendance

We have submitted the abstract of our work to the Fourth HBCU Conference, April 8-9, 1997, Cleveland, Ohio. We also prepared a poster paper format and presented that on the second day of the conference. The presentation focused on the then current distributed computing packages MPI, PVM [4], and CORBA (COmmon Object Request Broker Arthictecture). As will be briefly discussed in the next section, CORBA is a rapidly growing in use and popularity. Indeed, one key aspect of our presentation at the conference was pointing out the lack of support for Fortran based software. This primarily stems from the fact that Fortran is not an object-oriented language. We have also pointed out various alternatives such as C++ wrappers around Fortran code so that tools like CORBA can be utilized. The last viewgraph of our presentation included some recommendations which is worth reiterating. The following is verbatim recap in a more concise format:

Why not a more radical approach?

- Redesign software based on Fortran 77 and Older along object-oriented principles
- Translate the source code to an object-oriented programming language (C++ or Java)

Biting the Bullet: Cons and Pros

Cons:
- Far greater price to pay for "going distributed"
- Performance (Fortran code probably runs faster)
- Physics and engineering-oriented NASA staff's close familiarity with Fortran is foregone

Pros:
- Redesign and reworking will improve the quality
- Much easier to provide graphics-based user interfaces
- Going distributed is much easier
- Clear goals for and better utilization of grantees..

3. Trends in Distributed Computing

CORBA, which has been discussed in conjunction with the NPSS design goals of NPSS in [5], was featured in a recent issue of Component Strategies (formerly known as Object Magazine [6]. The article discusses the recent work of Object Management Group(OMG) in extending CORBA's scope with respect to Java interface and support for Ada. It also discusses plans for CDL (Component Definition Language) which augments its IDL (Interface Definition Language). The article concludes with CORBA products becoming more robust and enjoying wider use in the future.

This month's issue of Communications of the ACM features CORBA on its cover and contain several articles [7-11]. Practically all aspects and future promise of this highly popular distributed computing standard are covered. An interesting aspect regarding CORBA's increasing popularity is that initially it was designed as a flexible tool with narrow scope. Perhaps one reason is simply the lack of other distributed computing tools. Another interesting aspect which is pointed out in article on CORBA 3.0 [9] is that CORBA can be regarded as an application integration tool. The same article discusses three significant new CORBA features: Portable Object Adapter.

From the foregoing, it appears that close attention should be paid to CORBA as a viable distributed computing and application integration tool. As for MPI and PVM which is not further discussed in this report, their scope is narrow, and covers only the parallel machines or local networks with distributed operating systems. By the distributed operating system, we mean something like Sun's NFS(Network File System, Solaris threading facilities, and the like.

4. Challenges and Remedies

4.1 Technical

All the Fortran based rocket engine design and simulation code that has been lent Hampton University under several software agreements as part of this (PRESS) project uses Unix platform. However, unfortunately, from the beginning of August 1998 until the end of September, our Unix SunOS based local area network has been inaccessible. At the end of September 1998, largely due to the needs of this project, a very limited access to only a single node of the local net was provided. There are no remote login, email, or even printing facilities. Remote login facilities naturally effect the implementation of MPI, as was pointed out. Meanwhile, the ACCL LACE cluster for which we have painstakingly obtained an account has not been doing so well either! Indeed, since July 1998, we have been unable to access that cluster, perhaps due to the account being discontinued.

4.2 Staffing

Unfortunately, for the entire third and final year of this project, we were unable to enlist the help of graduate student research assistants. This, despite the ample availability of funds in the budget, and for two reasons. One is the economy doing well and our graduate students having far more attractive alternatives. The other, which is more important, is that our graduate program has shrunk from 16 to only a few students.

The revised budget which was submitted as part of the one-year no-cost extension provides support for two graduate students with higher stipends and tuition allowance limits. However, as pointed out in the introduction, our internal development budget has been entered into the computer system a few days ago. Based on this, we are in a position to aggressively search for graduate student assistants. One plan we have is to enlist the help of a colleague, Robert Willis, in sending emails to other HBCUs.

4.3 Equipment and Software

As was pointed out in the very first interim report of this project, at the start of the project, we were not able to obtain a powerful personal computer platform. Instead, we ended up with two, 100 MHz desktop and 75 MHz laptop, with limited memory and hard disk unit capacity of 1.6 GB and .75 GB respectively. However, since Unix was the main platform for experimenting with Fortran based code, this did not much matter. However, if we are to move away from the Unix platform and base our work during the one-year extension period on PC platform, these may not be sufficient.

To this end, on the revised budget, \$ 5K for hardware and \$ 7K for software. Among items detailed are Lahey Fortran for PC platform and Visibroker (CORBA + Java) distributed computing tool. However, these funds could be applied to software tools serving the same or similar purpose. On the other hand, it is clear that the funds are not sufficient for obtaining such integrated design environment[13] tools as I-SIGHT. At this time, the only significant PC platform software we have is Borland's C++ 5.0 with Java extension (jvm 1.0, the oldest).

5. Conclusion

We have outlined our work in the last half of the funding period. We have shown how a demo package for RESSAP using MPI can be done. However, we also mentioned the difficulties with the UNIX platform. We have reiterated some of the suggestions made during the presentation of the progress of the at Fourth Annual HBCU Conference.

Although we have discussed, in some detail, how TURBDES/PUMPDES software can be run in parallel using MPI, at present, we are unable to experiment any further with either MPI or PVM. Due to X windows not being implemented, we are also not able to experiment further with XPVM, which it will be recalled, has a nice GUI interface. There are also some concerns, on our part, about MPI being an appropriate tool. The best thing about MPI is that it is public domain.

Although and plenty of documentation exists for the intricacies of using MPI, little information is available on its actual implementations. Other than very typical, somewhat contrived examples, such as Jacobi algorithm for solving Laplace's equation, there are few examples which can readily be applied to real situations, such as in our case.

In effect, the review of literature on both MPI and PVM, and there is a lot, indicate something similar to the enormous effort which was spent on LISP and LISP-like languages as tools for artificial intelligence research. During the development of a book on programming languages [12], when we searched the literature for very simple examples like taking averages, reading and writing records, multiplying matrices, etc., we could hardly find a any! Yet, so much was said and done on that topic in academic circles. It appears that we faced the same problem with MPI, where despite significant documentation, we could not find even a simple example which supports course-grain parallelism involving only a few processes.

From the foregoing, it appears that a new direction may be required for more productive research during the extension period (10/19/98 - 10/18/99). At the least, the research would need to be done on Windows 95/Windows NT based platforms. Moreover, with the acquisition of Lahey Fortran package for PC platform, and the existing Borland C++ 5.0, we can do work on C++ wrapper issues.

We have carefully studied the blueprint for Space Transportation Propulsion Integrated Design Environment for the next 25 years [13] and found the inclusion of HBCUs in that effort encouraging. Especially in the long period for which a map is provided, there is no doubt that HBCUs will grow and become better equipped to do meaningful research. In the shorter period, as was suggested in our presentation at the HBCU conference, some key decisions regarding the aging Fortran based software for rocket propellants will need to be made. One important issue is whether or not object oriented languages such as C++ or Java should be used for distributed computing. Whether or not "distributed computing" is necessary for the existing software is yet another, larger, question to be tackled with.

REFERENCES

[1] Huzel, D. K. and Huang, D. H. *Modern Engineering for Design of Liquid-Propellant Rocket Engines (Second Printing)*, AIAA, Inc., Washington, D.C. 20024, (1992).

[2] Sutton, G. P. *Rocket Propulsion Elements: An Introduction to Engineering of Rockets (6th Edition)*, John Wiley and Sons, Inc., New York, (1992).

[3] Snir, M., Otto, S.W., Huss-Lederman, S., Walker, D.W., Dongarra, J. *MPI: The Complete Reference*, The MIT Press, Cambridge, Massachusetts (2nd Printing, 1997).

[4] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., Sunderam, V. *PVM: Parallel Virtual Machine, A User's Guide and Tutorial for Networked Parallel Computing*, The MIT Press, Cambridge, Massachusetts (Third Printing, 1996)

[5] Williams, A., Follen, G., Claus, R., Blech, R, et. al. "Key Technologies for Implementing An NPSS," Technical Memorandum, NASA Lewis Research Center, February 1997, [DRAFT].

[6] Guttman, M. and Applebaum, R. "The Next Generation of CORBA," Component Strategies, August 1998, pp. 40-46.

[7] Seetharaman, K. "The CORBA Connection," Comm. of ACM, vol. 41, no. 10 (October 1998)

[8] Siegel, J. "CORBA and the OMA in Enterprise Computing," Ibid.

[9] Vinoski, S. "New Features for CORBA 3.0," Ibid.

[10] Schmidt, D. C. "Evaluating Architectures for Multithreaded Object Request Brokers," Ibid.

[11] Haggerty, P. and Seetharaman, K. "The Benefits of CORBA-based Network Management," Ibid.

[12] Cezzar, R. *A Guide to Programming Languages: An Overview and Comparison*, Artech House Publishers, Inc., Norwood Mass. 02062, 1995.

[13] Hemminger, J. A. Space Transportation Propulsion Integrated Design Environment (STP/IDE), presentation at NASA Lewis Research Center, July 29, 1998.

[14] Follen, G. Williams, A., Blech, R, and Drei, D.V. (NASA Lewis), Apel, A. (P&W East Hartford), Byrd, R. (P&W, West Palm Beach), Gardocki, M. (G.E. Aircraft Engines), Crawford, N. (AlliedSignal Engines), Ashleman, R. (Boeing), McNelly, M. (Allison Engine Co.) "Numerical Propolsion System Simulation Architecture Definition," NASA TM 107343, November 1996.

[15] Miller, B., Szuch, J. R., Gauigier, R. E., Wood, J.R. "A Perspective on Future Directions in Aerospace Propulsion System Simulation," Lewis Research Center, NASA TM 102038, 1989.

[16] Reese, D. S., and Luke, E., "Object Oriented Fortran for Development of Portable Parallel Programs, Mississippi State University NASA Grant NAG3-1073 and NSF cooperative agreement ECD-8907070).

[17] Blech, R.A. and Arpasi, D.J., "An Approach to Real-Time Simulation Using Parallel Processing,", NASA TM 81731, 1981.

[18] Jell, Thomas, "Building Multimedia Application Systems Using Component Technology," WEB APPS Solutions Report W4, Object Expo/JAVA Expo '97, New York City, New York, May 2, 1997 (presentation viewgraphs).

APPENDIX A: Authorization for No-cost Extension of NAG3-1792

National Aeronautics and
Space Administration

Lewis Research Center
Cleveland, OH 44135-3191

NASA

Reply to Attn of: 0612

July 7, 1998

Hampton University
Office of Sponsored Programs
Hampton, VA 23668

Subject: NAG3-1792, Supplement No. 3

In accordance with the clause of the grant entitled, "Extensions", the period of performance completion of the subject grant is hereby extended from October 19, 1998 to October 18, 1999.

This authorization is conditioned upon the completion of the grant objectives within the current level of funding.

This letter of authorization constitutes Supplement No. 3 to the subject grant.

[illegible]
Syreeta Stewart
Grants Officer

cc:
ONRRO/Atlanta

To: [illegible]

From: [illegible]

Return ☐

Keep or Toss ☐

Post-it® 7668 ©3M 1993

[illegible] 7/14/98

APPENDIX B: Internal Memorandum Concerning Spending Rate

HAMPTON UNIVERSITY
HAMPTON, VIRGINIA 23668

OFFICE OF THE VICE PRESIDENT FOR
BUSINESS AFFAIRS AND TREASURER
(757) 727-5213
FAX (757) 727-5084

MEMORANDUM

TO: Dr. Ruknet Cezzar
Computer Science Department

FROM: Leon L. Scott [illegible]
Vice President for Business
Affairs and Treasurer

DATE: July 15, 1998

RE: **Unexpended Grant Funds**

Our records indicated as of, **June 30, 1998** you have expended only 52 percent of your award which is due to expire on **October 18, 1999.** I am closely monitoring the spending rate of all grants. You must adjust your spending to meet the requirements of your award. Please work with the Grants Management Office to spend.

Grant NO.: **5-24048** Grant Description: **NASA-Lewis Parallelization of Rocket Engine**

Budget	Expenditures and Encumbrances	Balance Available	Percent Used
$293,050	$ 153,178	$ 139,872	52

Thank you for your cooperation.

cc: Dr. Calvin Lowe, Vice President for Research
Dr. JoAnn Haysbert, Assistant Provost
Dr. Johnnye Jones, Dean, School of Science

APPENDIX C: Acquisition of new GASP module for PUMPDES/TURBDES

National Aeronautics and
Space Administration
Lewis Research Center
Cleveland, OH 44135-3191

Reply to Attn of: 5880

December 22, 1997

Dr. Ruknet Cezzar, Associate Professor
Department of Computer Science
Science and Technology Bldg, Room 120
Hampton University
Hampton, VA 23668

Dear Dr. Cezzar:

Per the recently signed Software Use Agreements, enclosed you will find three 3.5-inch diskettes containing the Fortran source code for the GASP (1 diskete) and GASPlus (2 diskettes) software.

Note the following:

(1) There are manuals available for both packages. However, we have no spare copies at this time, so it will take a while to get copies made. Let us know if you want one or both of them.

(2) The GASP source code is that used by Dean Scheer when he developed the PUMPDES and TURBDES codes for which you want to use it. Bob Hendricks inserted some lines up front to print out hydrogen property tables. You will have to remove these lines and make whatever modifications are required to get it to run with the noted codes.

(3) The GASPlus code is configured to run in a DOS-based, stand-alone mode. You can use it to check out the results obtained with GASP software.

I hope you find both of these software routines useful in carrying out your work under NASA Grant NAG3-1792.

Sincerely,

Joseph A. Hemminger

Joseph A. Hemminger
Engine Systems Technology Branch
Turbomachinery & Propulsion Systems

3 Enclosures

APPENDIX D: Acquisition of Miscellanous Modules Including RESSAP

National Aeronautics and
Space Administration
Lewis Research Center
Cleveland, OH 44135-3191

Reply to Attn of: 5880

MAY 18 1998

Dr. Ruknet Cessar, Associate Professor
Department of Computer Science
Hampton University
Science and Technology Building
Room 120
Hampton, VA 23668

Dear Dr. Cezzar:

Enclosed you will find three 3.5" floppy diskettes containing archival files from Dean Scheer's work on the Pumpdes and Turbdes Codes which have already been delivered to you.

They include:

1. Documentation files (in the TPDESDOC folder; for Pumpdes, Turbdes, and other routines; Word Perfect format; readable in MS Word)

2. Fortran source files (in the SOURCCDS folder)

3. Miscellaneous supporting files from his VAX account (in the MISCVAXF folder).

I hope you find them useful in carrying out the work under NASA Grant NAG3-1792.

Joseph A Hemminger

Joseph A. Hemminger
Engine Systems Technology Branch
Turbomachinery and Propulsion Systems Division

3 Enclosures

APPENDIX E: A Typical Debugging Trace for A RESSAP Run

```
ressap: returned from READFILE a090
ressap: returned from ENTHALPYLEVEL a128
ressap: returned from THEORISP a130
ressap: returned from STAGTEMP a132
ressap: returned from CSSTARSUB a134
ressap: returned from TCAPD a146
Hello I'm pumppd!
pumppd: returned from SETUP a270
pumppd: Parameters passed to GASP:
  1   15     42.400000000000     168.00000000000    0.    0.   0
pumppd: returned from GASP a276
pumppd: returned from GASP a283
pumppd: returned from GASP a319
pumppd: returned from TABLE a374
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a49   Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  1
pumppd: returned from NEWTON a491  Flag KFLG10 =  3
Line 582:    0.24242338831148    0.35570149921394   0.
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
```

APPENDIX E: A Typical Debugging Trace for A RESSAP Run (Cont'd)

```
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   1
pumppd: returned from GASP a555
pumppd: returned from NEWTON a557  Flag KFLG20 =   3
pumppd: returned from GASP s570
pumppd: returned from LEAK s579
pumppd: returned from GASP s595
pumppd: returned from GASP s606
Line 713:   0.   0.   0.
Line 735:     1.0000000000000    0.60876135403851    0.
pumppd: returned from BLDLG s661
FRICFB =     9.3778428720395D-02
BNDRAD=     -4.6722675622228
HYDDBN =     9.2030735557570D-02
pumppd: calling PMPHD 729
pumppd: returned from PMPHD s726
pumppd: returned from GASP s775
pumppd:  returning to ressap!
ressap: returned from PUMPPD a168
ressap: returned from FANNO a178
ressap: returned from SGASP s199
ressap: returned from FANNO s226
ressap: returned from TURBPD s244
ressap: returned from PUMPPD s270
ressap: returned from FANNO s283
ressap: returned from NEWTON s300
ressap: returned from TURBPD s324
ressap: returned from FANNO s333
ressap: returned from FANNO s337
ressap: returned from SGASP s349
ressap: returned from FANNO s358
ressap: returned from NEWTON s374
Go to flag 375   0
Successful completion of calls!
```

APPENDIX F: Parts of RESSAP Source Code Relevant to MPI

```
C*****************  MPI related declarations *****************
    parameter (control_id=0)
C  Control-id is the id of the process, in this case process 0

        CHARACTER*20 s_msg
        INTEGER i_msg, myrank, ierr, status(MPI_STATUS_SIZE)
C  Note: MPI_STATUS_SIZE is system implementation dependent

        REAL*8  r_msg
c
c  Global variables for ressap.for
c
    COMMON    /globals/ my_id, nprocs
C  These are the process id and number of processes
C
C***************  End of MPI related declarations ************
C
C
C      ROCKET ENGINE SYSTEM PRELIMINARY DESIGN CODE
C
       IMPLICIT REAL*8(A-H,O-Z)
       REAL*8 MU,MUL,MUV,K,KL,KV,ISPGG,ISPTP,ISPLL,ISP,ISPACT,MACHGG,
     1 MACHTP,MACHLL,MACH,MIX,LEVEL1,LEVEL2,LEVEL3,LSTAR,LF,NSSH2P,
     2 NSH2P,NSSO2P,NSO2P,LPRIME,LN,ISPSV
       DIMENSION ISPGG(10,11,14),ISPTP(10,11,14),ISPLL(10,11,14),
     c       TSTAGGG(10,11),TSTAGTP(10,11),TSTAGLL(10,11),
     c       CSTARGG(10,11),CSTARTP(10,11),CSTARLL(10,11),
     c       CPGG(10,11,15),CPTP(10,11,15),CPLL(10,11,15),
     c       GAMMAGG(10,11,15),GAMMATP(10,11,15),GAMMALL(10,11,15),
     c       MACHGG(10,11,15),MACHTP(10,11,15),MACHLL(10,11,15),
     c       VISCGG(10,11,15),VISCTP(10,11,15),VISCLL(10,11,15),
     c       PRGG(10,11,15),PRTP(10,11,15),PRLL(10,11,15),
     c       PRES(10),MIX(11),RATIO1(15),RATIO2(14)
C
C      LABELED COMMON STATEMENTS FOR H2/O2 PERFORMANCE AND PROPERTIES
C
       COMMON/ISPTAB/ISPLL,ISPGG,ISPTP
       COMMON/TSTAGTAB/TSTAGLL,TSTAGGG,TSTAGTP
       COMMON/CSTARTAB/CSTARLL,CSTARGG,CSTARTP
       COMMON/CPTAB/CPLL,CPGG,CPTP
       COMMON/GAMMTAB/GAMMALL,GAMMAGG,GAMMATP
       COMMON/MACHTAB/MACHLL,MACHGG,MACHTP
       COMMON/VISCTAB/VISCLL,VISCGG,VISCTP
       COMMON/PRTAB/PRLL,PRGG,PRTP
       COMMON/GROUP1/PRES,MIX,RATIO1
       COMMON/GROUP2/RATIO2
       COMMON/ENTHALPY/LEVEL1,LEVEL2,LEVEL3,DELTAHMIX
C
C      LABELED COMMON STATEMENT FOR "GASP"
C
       COMMON/PROPTY/KU,KZ,DL,DV,HL,HV,S,SL,SV,CV,CVL,CVV,CP,CPL,CPV,
     1 GAMMA,GAMMAL,GAMMAV,C,CL,CVP,MU,MUL,MUV,K,KL,KV,SIGMA,
     2 EXCL,EXCV,EXCESK
C     Source code continued on the next page ===============>
```

```
C      Notes on DATA statements for hydrogen/oxygen performance
C      and property tables
C       PRES - pressure is in psia
C       MIX - mixture ratio (O/F)
C       RATIO1 - RATIO1 data are the natural logarithms of thrust
C                chamber contraction and nozzle expansion area ratios.
C               Contraction ratios are negative, expansion ratios are positive
C       RATIO2 - RATIO2 data are the natural logarithms of nozzle expansion
C                area ratios.
C
      DATA PRES/20.0,50.0,100.0,200.0,300.0,400.0,500.0,
     C          1000.0,2000.0,5000.0/
      DATA MIX/2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0,11.0,12.0/
      DATA RATIO1/-1.60944,-1.38629,-1.09861,-.69315,.0,.69315,
     C 1.09861,1.38629,1.79176,2.30295,3.40120,4.09434,4.60517,
     C 5.70378,6.39693/
      DATA RATIO2/.0,.69315,1.09861,1.38629,1.79176,2.30295,
     C 3.40120,4.09434,4.60517,5.70378,6.39693,6.90776,7.09008,
     C 7.31322/
C
      NAMELIST/DEFALT/THETAC,THETAN,EPSRNA,EPSRNE,EPSE,EPSC,LSTAR,LF,
     1 B2BLDH,BLDTKH,WRAPH2,HOVTH2,NSSH2P,STGSH2,BLDSH2,EPSH2,UTMH2P,
     2 INRLBH,RLABYH,INNLH,NLH,INCLRH,CLRLH,
     3 B2BLDO,BLDTKO,WRAPO2,HOVTO2,NSSO2P,STGSO2,BLDSO2,EPSO2,UTMO2P,
     4 INRLBO,RLABYO,INNLO,NLO,INCLRO,CLRLO,
     5 KTRBH,NSTGH,UTMH2T,AOVRDH,CNOHNH,ALPH2H,CLRH,
     6 KTRBO,NSTGO,UTMO2T,AOVRDO,CNOHNO,ALPH2O,CLRO
C
      NAMELIST/REINPT/FVAC,DPRES,DMIX,DRATIO,PINO2B,TINO2B,PINH2B,
     1 TINH2B,P1OPMP,T1OPMP,P1HPMP,T1HPMP,DPOIJF,DPHIJF,DPCOOL,QCOOL,
     2 RPMH2P,RPMO2P,WTBPF,WOTBPF,DPHCVF,DPPMOF,DPOCVF,ETADH2,ETADO2,
     3 THETAC,THETAN,EPSRNA,EPSRNE,EPSE,EPSC,LSTAR,LF,
     4 B2BLDH,BLDTKH,WRAPH2,HOVTH2,NSSH2P,STGSH2,BLDSH2,EPSH2,UTMH2P,
     5 INRLBH,RLABYH,INNLH,NLH,INCLRH,CLRLH,
     6 B2BLDO,BLDTKO,WRAPO2,HOVTO2,NSSO2P,STGSO2,BLDSO2,EPSO2,UTMO2P,
     7 INRLBO,RLABYO,INNLO,NLO,INCLRO,CLRLO,
     8 KTRBH,NSTGH,UTMH2T,AOVRDH,CNOHNH,ALPH2H,CLRH,
     9 KTRBO,NSTGO,UTMO2T,AOVRDO,CNOHNO,ALPH2O,CLRO

C************************* MPI related calls *************************
C
      call MPI_INIT (ierr)
      call MPI_COMM_SIZE (MPI_COMM_WORLD, nprocs, ierr)
      call MPI_COMM_RANK (MPI_COMM_WORLD, my_id, ierr)
C     gets number of procecess and its id, it can now
C     compare my_id with control_id if need be
C
C********************* End of MPI related calls *********************
C
```

```
C      HYDROGEN PUMP DRIVE TURBINE SIZING AND PERFORMANCE
C

C*******  This call is replace by MPI message sent to TURBPD ********C
C                                                                    C
C      CALL TURBPD(KFLUID,KTRBH,KINPT,NSTGH,P1HTRB,T1HTRB,HPHPMP,    C
C     1 WTRBH,RPMH2P,UTMH2T,AOVRDH,CNOHNH,ALPH2H,CLRH,HPTRBH,DMTRBH, C
C     2 UMTRBH,UCTRBH,ETATSH,ETATTH,H1HTRB,D1HTRB,P2HTRB,T2HTRB,     C
C     3 H2HTRB,D2HTRB,S2HTRB,KTYPEH,ANSQDH,ADMH,Z1H,CP1H,GAM1H,      C
C     4 UTTRBH,HTR1H,HTR2H)                                          C
C                                                                    C
C************* end of replaced subroutine call *********************C
C
C************************  MPI related calls ************************
C
C     prepare the message as one integer value, and process id = 2
      i_msg = KFLUID
      id_proc = 2
C     This sets the receiving process id to 2, which is TURBPD
      i_tag = 1
C     This sets the position of the argument passed to TURBPD
C
      call MPI_SEND (i_msg, 1, MPI_INTEGER,
     *                   id_proc, i_tag, MPI_COMM_WORLD, ierr)
C
C     this sends the first argument, KFLUID, as the message to the
C     process TURBPD whose process id is 2
C     there are various alternatives, none too attractive (!), for
C     sending all other argument values to the TURBPD process
C     see the discussion in the Section 2.4 for more information
C
      call MPI_RECV (s_msg, 20, MPI_CHARACTER,
     *                   0, i_tag, MPI_COMM_WORLD, status, ierr)
C     receive a 20-char message, perhaps as an acknowledgment
C
C*********************  End of MPI related calls ******************
C
C    NOTE:  The rest of the source code for RESSAP
C                 ...
C
C    Any other call to TURBPD will be replaced by a message send in
C    the same way as was done for this call
C                 ...
C
C************************  MPI related calls ************************

     call MPI_FINALIZE(ierr)

C*********************  End of MPI related calls ********************
      STOP
      END
```