NASA/CR-2001-211250

Parallelization of Program to Optimize Simulated Trajectories (POST3D)

Dana P. Hammond
Raytheon Technical Services Company, Hampton, Virginia

November 2001

The NASA STI Program Office ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at (301) 621-0134

- Phone the NASA STI Help Desk at (301) 621-0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

NASA/CR-2001-211250

Parallelization of Program to Optimize Simulated Trajectories (POST3D)

Dana P. Hammond
Raytheon Technical Services Company, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under GSA Contract GS-00T-99-AKD-0209
NASA Task Order L-70750D

November 2001

Available from:

NASA Center for AeroSpace Information (CASI)
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 605-6000

Abstract

This paper describes the parallelization of the Program to Optimize Simulated Trajectories (POST3D). POST3D uses a gradient-based optimization algorithm that reaches an optimum design point by moving from one design point to the next. The gradient calculations required to complete the optimization process dominate the computational time and have been parallelized using a Single Program Multiple Data (SPMD) approach on a distributed-memory, non-uniform memory access (NUMA) architecture, namely the Origin2000.

Introduction

The following is the result of NASA's request to design and implement a parallel version of the analysis code, POST, to be used in the Reusable Launch Vehicle (RLV) low fidelity multidisciplinary analysis process. Initially, an analysis of the sample test cases was performed followed by an analysis of an RLV example. Based on the latter, a parallel implementation of the gradient calculations was developed and verified on an Origin2000.

Initial Analysis Based on Sample Test Cases

An in-depth analysis of POST3D in terms of parallel approaches was started, and the finite difference gradient calculations were identified as dominating the computational time central in completing the POST3D optimization. Either a finite differencing or an analytical method is used to compute derivatives. Both ways should be conducive to separating the gradient calculations with respect to design variables

```
del_OBJ/del_DV(i), del_G(j)/del_DV(i)

where: del_OBJ= derivative of objective function,
del_G(j) = derivatives of constraints,
del_DV(i) = derivatives design variables.
```

After computations of gradients they can be reassembled into the form required by the gradient based optimizer of choice (NPSOL, etc.). For the limited test cases, the **finite-differencing gradient calculations** appear to account for about 50% of the total CPU time; which limits the maximum achievable speedup.

The finite difference gradient calculations dominate the computational time central in completing the POST3D optimization. Using the grof (Appendix B) of Sample 2, the least intrusive locations to insert and coordinate parallelization is in gradient calculations, i.e., the gradients to each of the targets and to the optimization index with respect to the controls. If the search mode is 6 (stanford npsol), then gradnps.f in performs the calculations, else for all other search modes (4: projected gradient method, 5: accelerated projected gradient method) grad.f performs the calculations.

As a preliminary validation that the gradient calculations are independent and are candidates for parallelization, the independent variable loop (see do 300 in Appendix C) in both grad.f and gradnps.f were reversed. The results were validated to be consistent. A further analysis (discussion with program author) is required to ensure no boundary condition information is being saved in the common blocks.

To determine the amount of time required for the gradient calculations, a CPU timer was inserted prior to the gradient calculations performed for all the independent variables (nindv) and a CPU timer was inserted after the calculations. It should be noted that the actual times reported would vary based on the architecture and CPU speed. The relative CPU times between the total and gradient times are the only significant results being presented.

The three examples test cases provided with the POST3D Utilization Manual [4] were used for evaluation.

	Total CPU Time	Gradient CPU Time	Gradient/Total(%)
Sample 1	10.757	4.936	45.886
Sample 2	99.275	52.976	53.362
Sample 3	35.068	19.209	54.776

The independent variable loop (do 300) in both grad.f and gradnps.f forms a natural boundary for the distribution of the computation across processors, but limits the maximum number of processors to the number of independent variables. To maximize load balancing, the ideal situation is to evenly divide the independent variable gradient calculations to processors.

This parallelization approach provides the greatest reduction in total CPU when the number of gradient calculation increase and the number of independent variables increase. Unfortunately for the examples above, a large portion of the code (non-gradient calculation) is serial in nature and limits the projected CPU speedup. Even with ideal load balancing, **Amdahl's Law** projects the maximum achievable speedup (S) by a parallel algorithm with (P) processors given a percentage of serial work (F):

```
S <= 1 / (F + (1-F) / P)
```

The following are the maximum achievable speedup for various number processors, where the percentage of serial work is 50% (roughly those shown for the POST3D examples):

Processors	Speedup
4	1.6
8	1.777
12	1.846
16	1.882

The communication overhead to pass the information to and from the gradient calculations (information scattered and gathered) will additionally impact the maximum achievable speedup.

Analysis of POST3D Based on a Representative RLV Problem

The three examples test cases provided with the POST3D Utilization Manual were used for the initial evaluation. However, based on the limited potential speedup of the gradient calculations an additional test case with 31 independent variables was obtained. The additional test case is a sample **space shuttle ascent trajectory (ov-102, 36000 lb p/l)**, and is denoted as "SSAT1" below. SSAT1 is a better candidate for parallelization, as its gradient calculations require substantially more CPU time.

	Total CPU Time	Gradient CPU Time	Gradient/Total (%)
Sample 1	10.757	4.936	45.886
Sample 2	99.275	52.976	53.362
Sample 3	35.068	19.209	54.776
SSAT1 (FFD)	768.078	736.512	95.890
SSAT1 (CFD	858.695	837.730	97.558
SSAT1 (PERTS)	784.353	744.486	94.917

For SSAT1 (FFD - Forward Finite Difference), the gradient calculations were called 21 times, and each of the 31 independent variables took about 1.13 seconds accounting for the Gradient CPU time above.

For SSAT1 (CFD - Central Finite Differences), the gradient calculations were called 12 times, and each of the 31 independent variables took about 2.235 seconds accounting for the Gradient CPU time above.

The **SSAT1 (PERTS)** refers to "Automatic PERTS under NPSOL control." The gradient calculations (npfd.f) were called 16 times and each of the 31 independent variables took about 1.5 seconds. An execution profile appears in Appendix D, and shows that any missing gradient calculations are performed in **npfd** and may be the focus of similar parallelization.

The POST3D author indicates that the projected gradient methods work well with problems having independent variables up to approximately 20 to 30. The **npsol** works well for problems with approximately 75 to 80 independent variables. Thus, in the near-term the largest expected speedup will be limited to about 75 independent variables.

The following is the **maximum achievable speedup** for various numbers of processors, where the percentage of **serial work is 95%** (roughly those shown for the POST3D SSAT1 examples using finite differences and NPSOL/PERTS):

Processors	Speedup	
4	3.478	
8	5.925	
12	7.742	(*)
16	9.143	
24	11.163	(*)
31	12.40	
32	12.550	(*)

The maximum number of computational processors for SSAT1 is 31 (i.e., the number of independent variables). Also, as denoted with an (*), not all Speedups are achievable because not all processors would have computations to perform. For example with 24 processors, each processor would calculate one set of gradient calculations, and then there would only be 7 independent variables (25 thru 31 inclusive calculations). Twenty-four processors would set idle while 7 processors would perform a second set of gradient calculations. Thus, the **maximum achievable, load balanced projections** for SSAT1 would be:

Processors	**Speedup**
4	3.478
8	5.925
9-15	5.925
16	9.143
17-30	9.143
31	12.400

Varying the Number of Independent Variables for SSAT1 (FFD)

An attempt to characterize performance by varying the number of independent variables only proved unsuccessful for SSAT1 (FFD). Changing the NINDV (number of independent variables) created the following table in the input stream.

Ind. Vars.	**Total CPU Time**	**Gradient CPU Time**	**Gradient/Total (%)**
3	31.514	18.842	59.789
4	69.063	50.246	72.754
5	Trajectories Failed		
8	Trajectories Failed		
16	Trajectories Failed		

Coding Considerations

The POST3D author indicates that an effort in underway by another contractor to replace the common blocks in POST3D with structures. This version of the code is preliminary and not available at this time. Ideally in terms of parallelization, the array involved in the gradient calculations should exhibit unit-stride for optimal execution performance.

Implementation Approach

The parallelization approach of POST3D is classified as **Single Program Multiple Data (SPMD)** onto distributed memory **NUMA (non-uniform memory access)** architecture. The same program would be distributed to multiple processors communicating through a communication library. Each process would be part of a group and have a unique identification within the group. In this scenario, a control node would serve as central point of contact and read the input file(s) and distribute the information to the compute nodes to perform their subset calculations.

A typical implementation approach will have one processor read the input and pass values to the various processors that compute a portion of the calculation. The POST3D gradient calculations comprise a large portion of the code. These gradient calculations use large COMMON Blocks. Thus in the approach implemented, each processor reads the program input and computes to the point of the gradient calculations.

As shown in Figure 1 below, each processor calculates its portion of the gradient calculations based on its processor ID. A call to MPI_Pack is made to pack the partial results. The packed message (i.e., subset of gradient calculations) is sent to the control processor.

Once the control processor completes its share of gradient calculations, it receives the partial gradient results from the other processors, calls MPI_Unpack and merges them into a collected result. The control processor then calls MPI_Pack and broadcasts the collected result to all processors. All processors receive the broadcast and call MPI_Unpack to update the arrays associated with the gradient calculations, then continue with program execution.

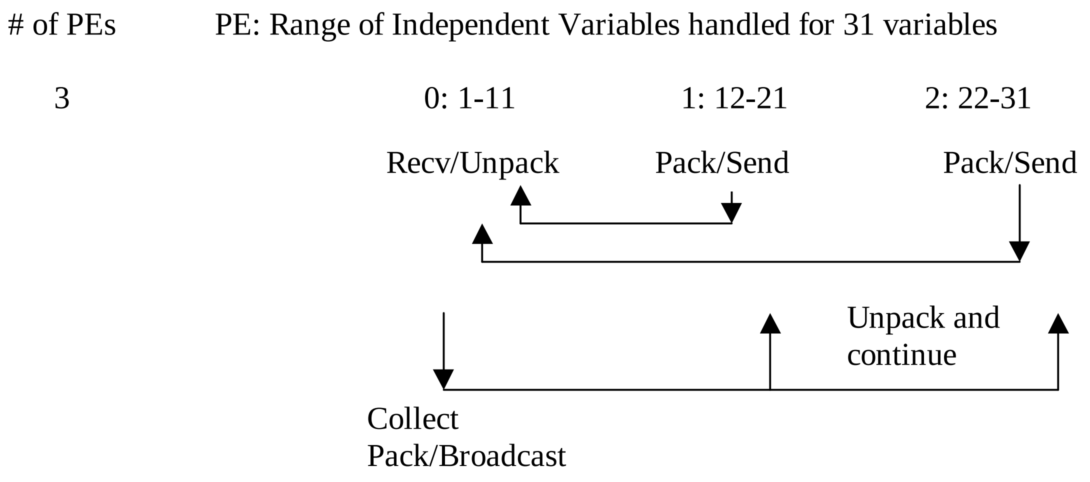

Figure 1. Communication between Processors

Implementation Details

A major consideration in the parallelization of POST3D was to minimize the amount of changes to existing code. As such, the bulk of the changes have been isolated into two routines: post3db/master.f and npsol/npfd.f. Additionally, instead of combining the serial and parallel versions into a single routine, and controlling which version to build by C preprocessor (ifdef) statements, a separate version of the affected routines were created. A similar approach was used for the makefiles.

Description of npfd_par.f

The routine npfd_par.f contains the control loop that performs the gradient calculations. The affected variables that take part in the calculations, appear to be contained in the arguments to npfd. However, depending on the number of processors, the subset of the arrays computed by the processor varies and therefore must be calculated and executed.

```
      > c dana: determine the loop start and end for the processor
      > c       (divisor, remainder, processor start/end element)
      >       idanaEle=n/numprocs
      >       idanaRem=n-(idanaEle*numprocs)
      >       if(idanaRem.gt.myid)idanaEle=idanaEle+1
      >       idanaStart=(myid*idanaEle)+1
      >       if(myid.ge.idanaRem)idanaStart=idanaStart+idanaRem
      >       idanaEnd=idanaStart+(idanaEle-1)
   >  do 340 j  = idanastart,idanaend
```

Once the subset of calculations has been performed, a call to a newly added subordinate routine (npfdio.F) is made to isolate the message passing operations.

```
    > C Call the message passing operations
    > C Note: the MPI calls and calculation were separated both to
    > C       minimize the code modification and ability to compile
    > C       with different options (POST3D requires the -static option)
    >   341 continue
    >       idanact=idanact+1
    >       call npfdio(idanakj,kki,n,ncnln,ldcj,ldcju,
    >      . bl,bu,grad,gradu,hforwd,hcntrl,x,
    >      . inform,bigbnd,cdint,fdint,fdnorm,objf,iprt0,icnfun,
    >      . c0,c1,c2,needc,
    >      . cjac,cjacu)
```

Description of npfdio.f

The exchange of gradient calculation information is performed in this routine. Logic to distinguish between the master and compute nodes and the necessary message passing exchanges is contained in npfdio.F. The master processor posts a MPI_RECV for each processor and waits until all partial gradient calculations have been received.

In order to minimize the number of messages sent, i.e., one for each array involved in the gradient calculation, the MPI_PACK and MPI_UNPACK routines were used to consolidate arrays. The number of array elements to be packed and the location of the elements within the array must be calculated based on the processor from which the calculations were performed. For example, in the code below the compute node packs **kis** elements from the **bl** array starting at location **kki**. The values of **kis** and **kki** are calculated based on the number of processors and the compute node's processor id.

```
c Pack
c   -------------------
      idanaEle=n/numprocs
      idanaRem=n-(idanaEle*numprocs)
      if(idanaRem.gt.myid)idanaEle=idanaEle+1
      idanaStart=(myid*idanaEle)+1
      if(myid.ge.idanaRem)idanaStart=idanaStart+idanaRem
      idanaEnd=idanaStart+(idanaEle-1)

kki=idanaStart
      kis=idanaEle
c   --------------------
      iposition=0

      call MPI_PACK(bl(kki),kis,MPI_DOUBLE_PRECISION,
       *          ibytes,ibytesize*4,iposition,MPI_COMM_WORLD,impierr)
```

Description of master_par.f

The routine master_par.f must initialize MPI and enroll all the compute nodes. Each processor will read the input files and potentially write output files, which may be rewound and used during computation, therefore each processor must control its own data files to ensure data integrity. Finally MPI is terminated gracefully.

Implementation Considerations

Currently, only synchronous message passing has been implemented [1: *Using MPI*, William Gropp]. Deferred synchronization [2: *Using MPI-2*, William Gropp] could readily be implemented using MPI_IRECV and MPI_WAITSOME for additional gains in performance.

A typical implementation approach is to have one processor read the input, and pass values to the various processors, which compute a portion of the calculation. The POST3D gradient calculations consist of a large portion of the code, and make significant use numerous and large COMMON blocks. The depth of the routines called in the gradient calculations (call-tree), together with the large number of COMMON blocks, precludes an analytical validation of the parallel approach. The parallel approach is valid if each independent variable's gradient calculations are fully exchanged in the message-passing approach. There can be no implicit exchange of information between independent variable though COMMON blocks by subordinate routines.

```
do 340 j = 1, number_of_independent_variables

           % Time         Time       # of Calls          Routine

   [12]        52.5        389.33       361             confun_ [12]
   [16]        22.5        167.22       516             objfun_ [16]

   that call

   [6]         75.7        561.79       521             traj_ [6]
   [8]         74.1        549.26       1563            phzxm_ [8]
   [9]         73.1        537.14       268836          ruk_ [9]
   [10]        72.7        518.16       1082638         motion_ [10]
```

```
[14]        33.9        236.24         1082638          auxfm_ [14]
                          7.64    218692876/324754409   gentab_ [19]
                         29.52    1082638/1082638       georate_ [26]
                         17.66    1082638/1082638       prop_ [27]
                         22.32    1082638/1082638       aero_ [28]
                         18.88    1082638/1082638       tmotm_ [34]
                         11.64    1082638/1082638       gdgclt_ [36]
                         12.75    1082638/4331073       atmos_ [23]
```

Summary of Results

An example, **space shuttle ascent trajectory, ov-102, 36000 lb p/l,** denoted as SSAT1, was provided with 31 independent variables.

The following is the **maximum achievable speedup** for various numbers of processors, where the percentage of **parallel work is 95%** (roughly those shown by SSAT1):

Processors	Speedup
2	1.905
3	2.727
4	3.478
8-15	5.925 *
16-30	9.143 *
31	12.400

(* Idle processors. Independent variables cannot be divided equally among processors.)

Initial Timing Results using Origin2000 (whitcomb)

PBS, mpich-1.2.1, 64bit, IRIX64 whitcomb 6.5

16 250 MHZ IP27 Processors

CPU: MIPS R10000 Processor Chip Revision: 3.4

FPU: MIPS R10010 Floating Point Chip Revision: 0.0

Main memory size: 16384 Mbytes

f77 -col72 -DSGI -r10000 -mips4 -64 -O2 -c

CPU Time is derived from dtime (same as used by POST3D)

# of PEs	**Actual CPU (wall)**	**Projected**	**(Serial time/Speedup)**
1	164	(165)	-
2	94,93	(97)	86 (164/1.905)
3	68,67,67	(70-74)	60
4	53,52,50,51	(56)	47
5	49,44,43,43,45	(51)	
6	43,38,…,38	(45)	
8	34,32,…,31	(38)	28
10	34,29…,29	(40)	
16	31,30,…,24,22	(47)	18

Analysis of Results

As the number of processors increase, the corresponding CPU time required for the gradient calculations decrease as expected. The wall clock time however, appears to scale to about eight processors then begins to increase. The lack of scalability is due to the synchronous communication costs. To correct this, deferred synchronization [2: *Using MPI-2*, William Gropp] could readily be implemented using MPI_IRECV and MPI_WAITSOME for additional gains in performance. This would reduce the serialization of the messages being received by the master processor.

Additionally, the current implementation passes messages from all the compute processors directly to the master process; this is an order (n) approach. An order log (n) algorithm could be implemented in which the processors pass their contributions to neighbors in a binary b-tree approach, and eventually to the master processor. This would reduce the dependency of one processor receiving all the messages.

Summary of Code Changes

The parallel version of POST3D has been implemented on an Origin2000 (SGI) and cluster of Sun workstations. The POST3D base codes provided for these architectures were different, reflecting system dependencies. However the parallel implementation affected a common subset of subroutines, and was therefore the same.

The following additional files have been added to the parallel version (i.e., the serial version remains unchanged).

```
inc/postmpi.inc       /* include file for MPI related information */
post3db/master_par.f  /* modified the I/O for master process */
post3db/Makefile_par  /* makefile to compile parallel version of
                         master_par.f */
npsol/npfd_par.f      /* modified NPSOL gradient calculation */
npsol/Makefile_par    /* makefile to compile parallel version of npfd.f */
npsol/npfdio.F        /* the message passing was decoupled from the
                         calculations */
exe/Makefile_par      /* makefile to create the parallel execution */
```

How to Compile and Execute the Parallel Version of POST3D

The parallel version of POST3D has been implemented on an Origin2000 (SGI) and cluster of Sun workstations**.** The compilation process has been encapsulated by makefiles such that the compilation is the same for both machines. It is assumed that the reader knows how to link in the required MPI library.

To make a serial version (creates **exe/post**):

```
./makefile.exe
```

To make the parallel version of POST3D (creates **exe/post_par**):

```
./makefile_par.exe
```

The environment differs between these two architectures when running MPI codes. Below is a description of how to execute in each environment. The input cases usually reside in the inputs directory. A subdirectory, called Big1, contains the SSAT1 example. A POST3D execution requires at least two files residing in the execution directory: input and npinput. The input file must be called “input.”

Origin2000

In the tar file provided as part of the Origin2000 delivery is in the **inputs/Big1** directory. The origin2000 on which POST3D was executed used the Portable Batch System (PBS). To submit a job, the user uses the qsub command to describe the resource (i.e., wall time, number of cpus, etc.). Here is an example.

To run the serial version:

```
cd inputs/Big1

   ../../exe/post < input > tout
```

To run the parallel version in the batch environment:

```
qsub -l walltime=20:00,ncpus=2 ./pbsjob2

  where 20 minutes was requested for 2 cpus.
```

The job to be executed is contained in pbsjob2.

```
whitcomb> more pbsjob2
#PBS -m e
cd $PBS_O_WORKDIR
cd inputs/Big1
mpirun -np 2 ../../exe/post_par < input > tout
whitcomb>
```

This will generate at least the following files:

```
  profila profilb npost3d.out npost3d.rst summary tout
```

To compare results:

```
  diff tout ../Gold
```

These files must be deleted before the next run; else they may conflict with the creation of new files.

<u>Cluster of Sun Workstations (MPICH 1.2.1)</u>

MPICH 1.2.1 is the version of MPI used for message passing on the cluster of workstations. For this installation, MPICH was installed in my area, but typically the system administrator should install it in a public area. Note, because the cluster does not have a batch system, a call to mpirun is all that is required.

To run the serial version:

```
   cd inputs/Big1
      ../../exe/post < input > tout
```

To run the <u>parallel</u> version:

```
   cd inputs/Big1
      ~/mpich-1.2.1/bin/mpirun -np 2 ../../exe/post_par < input > tout
```

Again, this will generate at least the following files:

```
  profila profilb npost3d.out npost3d.rst summary tout
```

These files must be deleted before the next run; else they may conflict with the creation of new files.

<u>Future Work</u>

There are several outstanding work items that could be valuable, but were not pursued due to the concerns with the budget constraints. These items could readily be completed upon request.

The current version of the gradient calculations using NPSOL (analytical) is implemented with **synchronous** communication. Asynchronous (or **deferred synchronous** communication) would probably result in a must scalable code (i.e., greater than 8 to 12 processors).

The **projected gradient** derivatives, using finite differencing, may benefit from parallelization when used with large number of independent variables. The program author indicated that 20 to 30 independent variables were the mathematical constraints. Thus, if several test cases could be provided for these gradient methods having the upper end of independent variables, parallelization may be demonstrated.

Finally, the next version of POST3D, named "POST II," may be available to examination by summer. The major change between the two versions is namelist and the data structures within the program. The parallel algorithm implemented in POST I should readily be instrumented in **POST II**. Additionally, POST II has been extended to support multiple launch vehicles. It is believed that "**coarse grain**" parallelism could be applied, with the division of work segmented at the vehicle level. It may be possible to combine the fine grain parallelism of POST I with the coarse grain parallelism of POST II for even better performance.

Bibliography

Gropp, William; Lusk, Ewing; and Skjellum, Anthony: *Using MPI. Portable Parallel Programming with the Message-Passing Interface*. MIT Press, 1994.

Gropp, William; Lusk Ewing; and Thakur, Rajeev: *Using MPI-2, Advanced Features of the Message-Passing Interface.* MIT Press, 1999.

Powell, R. W.; Striepe, S. A.; Desai, P. N.; and Braun, R. D.: *Program To Optimize Simulated Trajectories (POST*), Volume II, Utilization Manual, (Version 5.2), October 1997

Venter, Gerhard; and Watson, Brian: *Efficient Optimization Algorithms for Parallel Applications*, AIAA-2000-4819, Vanderplaats Research and Development, Inc.

Appendix A

Gprof of Sample2

more sample2_dana_pg.gprofcopy
…
granularity: each sample hit covers 2 byte(s) for 0.01% of 141.93 seconds

```
                              Called/Total      Parents
Index      %Time            Self Descendents  Called+Self          Name     Index
                              Called Total      Children

            0.00               110.88          1/1              _start [2]
[1] 78.1    0.00               110.88          1                main [1]
            0.00               110.88          1/1              MAIN_ [3]
            0.00                 0.00          1/1              __f77_init [368]
            0.00                 0.00          1/1              f77_init [533]

--------------------------------


                                                                <spontaneous>
[2] 78.1    0.00               110.88                           _start [2]
            0.00               110.88           1/1             main [1]
            0.00                 0.00           4/4             atexit [525]

--------------------------------

            0.00               110.88           1/1             main [1]
[3] 8.1     0.00               110.88           1               MAIN_ [3]
            0.00               110.08           1/1             tspxm_ [4]
            0.00                 0.72           1/1             readat_ [105]
            0.00                 0.04           1/1             _s_stop [223]
            0.00                 0.02           1/1             savdat_ [273]
            0.00                 0.01           1/1             __fdate_ [329]
            0.00                 0.01           5/12            __f_open_nv [271]
            0.00                 0.00           1/2             dacopn_ [363]
            0.00                 0.00           6/15             __f_rew [360]
            0.00                 0.00           1/2             second_ [379]
            0.00                 0.00           2/4052          __e_wsfe [111]
            0.00                 0.00           3/261           __f_clos [365]
            0.00                 0.00           1/4052          __s_wsFe_nv [285]
            0.00                 0.00           2/317           __do_l_out [394]
            0.00                 0.00           1/64            __s_wsle_nv [444]
            0.00                 0.00           1/64            __e_wsle [443]
            0.00                 0.00           1/355816        __s_copy [237]
            0.00                 0.00           1/1             __signal_ [1100]
            0.00                 0.00           1/1             usero_ [548]
            0.00                 0.00           1/1             exit [532]

--------------------------------
```

```
                        Called/Total       Parents
Index      %Time      Self Descendents   Called+Self       Name      Index
                        Called Total       Children

             0.00           110.08      1/1              MAIN_ [3]
[4] 77.6     0.00           110.08      1                tspxm_ [4]
             0.00           109.97      1/1              nlprg_ [5]
             0.08             0.00      1/1              nomtab_ [196]
             0.00             0.02      14928/2190872    __do_u_in [46]
             0.00             0.01      2/3              __s_rsue_nv [294]
             0.00             0.00      1/2              dacopn_ [363]
             0.00             0.00      1/12             __f_open_nv [271]
             0.00             0.00      1/2              second_ [379]
             0.00             0.00      1/15             __f_rew [360]
             0.00             0.00      1/4052           __e_wsfe [111]
             0.00             0.00      1/2207986        .div [89]
             0.00             0.00      4/10238          locf_ [468]
             0.00             0.00      2/3              __e_rsue [1090]

---------------------------------------------

             0.00           109.97      1/1              tspxm_ [4]
[5] 77.5     0.00           109.97      1                nlprg_ [5]
             0.00           108.03      1/1              npsol_ [8]
             0.00             1.89      1/58             cnfunc_ [6]
             0.00             0.02      2/2              nlout_ [252]
             0.00             0.01      1/1              opfile_ [283]
             0.00             0.01      1/1              npslic_ [326]
             0.00             0.00      31/86146         __do_f_out_nv [19]
             0.00             0.00      1/1              npoptn_ [374]
             0.00             0.00      6/4052           __e_wsfe [111]
             0.00             0.00      1/15             __f_rew [360]
             0.00             0.00      1/26             fflush [343]
             0.00             0.00      1/341            __e_wsfi [214]
             0.00             0.00      6/4052           __s_wsFe_nv [285]
             0.00             0.00      1/1              npfile_ [463]
             0.00             0.00      1/442            __c_fi [414]
             0.00             0.00      1/8982039        .mul [71]
             0.00             0.00      1/442            __c_si [1037]
             0.00             0.00      1/341            __s_wsFi_nv [1038]
             0.00             0.00      1/58             chkvec_ [495]
             0.00             0.00      1/1              npsloc_ [545]
             0.00             0.00      1/1              calwef_ [529]
             0.00             0.00      1/22             flush_ [509]

---------------------------------------------
```

```
                             Called/Total        Parents
Index        %Time         Self Descendents    Called+Self          Name       Index
                             Called Total        Children

              0.00               1.89          1/58             nlprg_ [5]
              0.00             107.65          57/58            confun_ [9]
[6] 77.2      0.00             109.54          58               cnfunc_ [6]
              0.00              87.44          57/57            gradnps_ [13]
              0.00              22.10          58/287           traj_ [7]
              0.00               0.00          2/740            pager_ [187]
              0.00               0.00          57/57            grad_ [498]

--------------------------------

              0.00              22.10          58/287           cnfunc_ [6]
              0.00              87.25          229/287          grad2nps_ [14]
[7] 77.0      0.00             109.35          287              traj_ [7]
              0.03              98.63          2299/2299        phzxm_ [12]
              0.00               4.69          286/286          setic_ [37]
              0.00               2.85          2299/2299        phzxmi_ [49]
              0.00               2.84          754/754          savic_ [50]
              0.00               0.18          2299/2299        clspfl_ [167]
              0.03               0.06          2013/2013        dinpt_ [191]
              0.03               0.00          2299/2299        setiv_ [235]
              0.00               0.00          6/86146          __do_f_out_nv [19]
              0.00               0.00          2/4052           __e_wsfe [111]
              0.00               0.00          2/740            pager_ [187]
              0.00               0.00          574/355816       __s_copy [237]
              0.00               0.00          2/4052           __s_wsFe_nv [285]
              0.00               0.00          4598/4598        calf_ [469]

--------------------------------

              0.00             108.03          1/1              nlprg_ [5]
[8] 76.1      0.00             108.03          1                npsol_ [8]
              0.00             106.10          1/1              npcore_ [10]
              0.00               1.89          1/1              npchkd_ [65]
              0.00               0.03          1/1              npdflt_ [247]
              0.00               0.01          1/19             nomout_ [181]
              0.00               0.00          1/1              cmchk_ [386]
              0.00               0.00          1/19             lscore_ [338]
              0.00               0.00          2/4052           __e_wsfe [111]
              0.00               0.00          1/86146          __do_f_out_nv [19]
              0.00               0.00          2/4052           __s_wsFe_nv [285]
              0.00               0.00          5/101            cmqmul_ [405]
              0.00               0.00          3/3810562        __pow [30]
              0.00               0.00          1/1              dgeqr_ [450]
              0.00               0.00          13/745           dcopy_ [403]
              0.00               0.00          6/232            dload_ [421]
              0.00               0.00          4/42             icopy_ [441]
```

```
                        Called/Total      Parents
Index      %Time      Self Descendents  Called+Self         Name     Index
                        Called Total      Children


             0.00               0.00       1/1              npploc_ [458]
             0.00               0.00       1/2              f06qhf_ [456]
             0.00               0.00       1/4              mchpar_ [455]
             0.00               0.00       1/43             f06qff_ [439]
             0.00               0.00       1/134            dcond_ [429]
             0.00               0.00       1/141            dscal_ [428]
             0.00               0.00       1/1              lscrsh_ [464]
             0.00               0.00       1/1              lsbnds_ [540]
             0.00               0.00       1/1              lssetx_ [541]
             0.00               0.00       1/1              npcrsh_ [544]

-------------------------------------------

             0.00               1.89       1/57             npchkd_ [65]
             0.00             105.77       56/57            npsrch_ [11]
[9] 75.8     0.00             107.65       57               confun_ [9]
             0.00             107.65       57/58            cnfunc_ [6]

-------------------------------------------

             0.00             106.10       1/1              npsol_ [8]
[10] 74.8    0.00             106.10       1                npcore_ [10]
             0.00             105.77       18/18            npsrch_ [11]
             0.00               0.18       18/18            npprt_ [165]
             0.00               0.13       18/19            nomout_ [181]
             0.01               0.01       18/18            npiqp_ [272]
             0.00               0.00       18/308           .rem [228]
             0.00               0.00       4/86146          __do_f_out_nv [19]
             0.00               0.00       1/20             cmprt_ [337]
             0.00               0.00       1/4052           __e_wsfe [111]
             0.00               0.00       17/17            npupdt_ [407]
             0.00               0.00       1/1              nprset_ [412]
             0.00               0.00       115/745          dcopy_ [403]
             0.00               0.00       17/101           cmqmul_ [405]
             0.00               0.00       18/18            npmrt_ [431]
             0.00               0.00       52/134           dcond_ [429]
             0.00               0.00       19/574           ddot_ [409]
             0.00               0.00       1/4052           __s_wsFe_nv [285]
             0.00               0.00       88/261           ddiv_ [459]
             0.00               0.00       2/355816         __s_copy [237]
             0.00               0.00       52/490           dnrm2_ [481]
             0.00               0.00       18/18            npfeas_ [514]
             0.00               0.00       18/18            npalf_ [513]

-------------------------------------------
```

```
                        Called/Total       Parents
Index      %Time      Self Descendents   Called+Self          Name    Index
                        Called Total       Children

                0.00          105.77     18/18                npcore_ [10]
[11] 74.5       0.00          105.77     18                   npsrch_ [11]
                0.00          105.77     56/57                confun_ [9]
                0.00            0.00     424/574              ddot_ [409]
                0.00            0.00     417/745              dcopy_ [403]
                0.00            0.00     292/470              daxpy_ [413]
                0.00            0.00     56/269               dgemv_ [399]
                0.00            0.00     112/130              ddscl_ [430]
                0.00            0.00     23/43                f06qff_ [439]
                0.00            0.00     18/19                iload_ [449]
                0.00            0.00     74/74                srchc_ [494]
                0.00            0.00     56/57                objfun_ [499]

-----------------------------------------------

                0.03           98.63     2299/2299            traj_ [7]
[12] 69.5       0.03           98.63     2299                 phzxm_ [12]
                0.67           83.50     63469/63469          ruk_ [16]
                0.07           11.50     68067/68067          infxm_ [24]
                0.09            1.42     4598/262774          motion_ [15]
                0.37            0.94     68067/68067          tgoem_ [79]
                0.05            0.00     68067/68067          cycxm_ [220]
                0.04            0.00     68067/68067          dynxm_ [233]
                0.00            0.00     4598/260475          deriv_ [158]
                0.00            0.00     63469/63469          dyns1_ [466]

-----------------------------------------------

                0.00           87.44       57/57              cnfunc_ [6]
[13] 61.6       0.00           87.44       57                 gradnps_ [13]
                0.00           87.38       229/229            grad2nps_ [14]
                0.00            0.06       228/684            pad_ [161]

-----------------------------------------------

                0.00           87.38       229/229            gradnps_ [13]
[14] 61.6       0.00           87.38       229                grad2nps_ [14]
                0.00           87.25       229/287            traj_ [7]
                0.00            0.13       456/684            pad_ [161]
                0.00            0.00       229/8982039        .mul [71]

-----------------------------------------------
```

Appendix B

Partial Code from gradnps.f

```
c....   start of do until ks >= nindv loop
c dana 12/04/00 reverse loop to verify independence
        print *,'gradnps:rev(1):',second(2)
        dana1=etimedif()
c       do 300 ks=1,nindv
        do 300 ks=nindv,1,-1
          sigdel = 0.0d0
          pertod = pert(ks)
c
c...      try a forward difference pass.
          call grad2nps(ks,0)
          call pad(pert(ks),u(ks),1)
          if ( isens.eq.1 .or. (sigdel+pdlmax).lt.0 ) then
c
c....        set pert to the negative of pert value before adjustment
            pertnw = pert(ks)
            pert(ks) = -pertod
            if ( prntpd.ne.0.0d0 ) then
              call pager (1)
              write (6,10030) ks,pert(ks)
10030         format ( ' reevaluate function with -pert(', i2,
     1        ') = ' , 1pe15.8, ' to get central differences' )
            endif
c
c....        save the forward error
c....        save forward p1 value
            if ( ndepv.ne.0 ) then
              do 80 l=1,ndepv
                esave(l) = depvl(l)
80            continue
            endif
            if ( opt.ne.0 ) then
              p1save = p1
            endif
c
c....        do a central difference pass.
            call grad2nps(ks,1)
c
c....        set pert to adjusted value for next iteration
            pert(ks) = pertnw
          endif
        print *,'gradnps:rev(2):',second(2)
300     continue
        dana2=etimedif()
        danatot=danatot+(dana2-dana1)
        print *,'gradnps:danatot=',danatot
```

…

Appendix C

Gprof of Space Shuttle Ascent Trajectory (SSAT1)

Space Shuttle Ascent Trajectory (SSAT1) is representative of POST3D problems where partial differentiation (ISENS) is computed by automatic PERTS under NPSOL control. This test case has 31 independent variables. One major difference/consequence is that gradient calculations are performed largely by **npfd.f**.

Highlights of profile: npsol [7] accounts for 75% of the program execution, the majority of which occurs in npfd [11] by means of npcore [13] and npchkd [15]. Specifically, 528.51 of 567.31 of execution time is spent in npfd (and its children) [11]. For this particular case, constraint functions (confun[12] = 341.76) required more than twice the execution time as the objective functions (objfun[16] = 160.74).

```
f77 -w -pg  -O3 -Nn4000 -Nl100 -Nq500 -c
…
granularity: each sample hit covers 2 byte(s) for 0.00% of 741.71 seconds

                        Called/Total      Parents
Index     %Time       Self Descendents  Called+Self         Name    Index
                        Called Total     Children

            0.00          567.31          1/1             _start [2]
[1] 76.5    0.00          567.31          1               main [1]
            0.00          567.31          1/1             MAIN_ [3]
            0.00            0.00          1/1             __f77_init [353]
            0.00            0.00          1/1             f77_init [518]

-----------------------------------------------
                                                          <spontaneous>
[2] 76.5    0.00          567.31          _start [2]
            0.00          567.31          1/1             main [1]
            0.00            0.00          4/4             atexit [511]

-----------------------------------------------

            0.00          567.31          1/1             main [1]
[3] 76.5    0.00          567.31          1               MAIN_ [3]
            0.00          565.54          1/1             tspxm_ [4]
            0.00            1.54          1/1             readat_ [117]
            0.00            0.12          1/1             __s_stop [201]
            0.00            0.05          5/12            __f_open_nv [204]
            0.00            0.03          1/1             savdat_ [267]
            0.00            0.02          1/1             __fdate_ [292]
            0.00            0.01          1/2             dacopn_ [295]
            0.00            0.00          2/2             etimedif_ [358]
            0.00            0.00          1/2             second_ [354]
            0.00            0.00          3/261           __f_clos [206]
```

```
                         Called/Total       Parents
Index     %Time        Self Descendents   Called+Self          Name      Index
                         Called Total       Children

             0.00            0.00        6/15             __f_rew [366]
             0.00            0.00        2/3541           __e_wsfe [153]
             0.00            0.00        5/10             __do_l_out [401]
             0.00            0.00        1/3541           __s_wsFe_nv [338]
             0.00            0.00        3/8              __s_wsle_nv [444]
             0.00            0.00        3/8              __e_wsle [443]
             0.00            0.00        3/366616         __s_copy [217]
             0.00            0.00        1/1              __signal_ [1097]
             0.00            0.00        1/1              usero_ [528]
             0.00            0.00        1/1              exit [517]

-----------------------------------------------

             0.00          565.54        1/1              MAIN_ [3]
[4] 76.2     0.00          565.54        1                tspxm_ [4]
             0.00          564.82        1/1              nlprg_ [5]
             0.67            0.00        1/1              nomtab_ [151]
             0.00            0.02      14928/3924656      __do_u_in [55]
             0.00            0.01        2/3              __s_rsue_nv [302]
             0.00            0.01        1/2              dacopn_ [295]
             0.00            0.01        1/12             __f_open_nv [204]
             0.00            0.00        1/2              second_ [354]
             0.00            0.00        1/3541           __e_wsfe [153]
             0.00            0.00        1/15             __f_rew [366]
             0.00            0.00        4/7286           locf_ [333]
             0.00            0.00        1/3940546        .div [114]
             0.00            0.00        2/3              __e_rsue [1090]

-----------------------------------------------

             0.00          564.82        1/1              tspxm_ [4]
[5] 76.2     0.00          564.82        1                nlprg_ [5]
             0.00          557.97        5/5              npsol_ [7]
             0.00            5.39        5/5              cnfunc_ [60]
             0.01            0.59        3207/25556       __do_f_out_nv [66]
             0.00            0.39        6/6              nlout_ [158]
             0.00            0.31        5/5              npslic_ [168]
             0.00            0.09        510/3541         __e_wsfe [153]
             0.00            0.03        4/4              art9_ [262]
             0.00            0.01        1/1              opfile_ [306]
             0.00            0.01        5/5              npoptn_ [316]
             0.00            0.00        1/1              npfile_ [364]
             0.00            0.00        10/36            fflush [350]
             0.00            0.00        506/3541         __s_wsFe_nv [338]
             0.00            0.00        5/416            __e_wsfi [227]
             0.00            0.00        5/1759           chkvec_ [195]
             0.00            0.00        1/15             __f_rew [366]
             0.00            0.00        781/8233966      .mul [124]
             0.00            0.00        5/10             __do_l_out [401]
             0.00            0.00        5/8              __s_wsle_nv [444]
                         Called/Total       Parents
Index     %Time        Self Descendents   Called+Self         Name       Index
                         Called Total       Children
```

```
            0.00            0.00       5/8              __e_wsle [443]
            0.00           0.00       10/366616        __s_copy [217]
            0.00           0.00       5/600            __c_fi [412]
            0.00           0.00       4/19             f06qhf_ [452]
            0.00           0.00       10/32            flush_ [487]
            0.00           0.00       5/600            __c_si [1032]
            0.00           0.00       5/416            __s_wsFi_nv [1036]
            0.00           0.00       5/5              npsloc_ [509]
            0.00           0.00       1/1              calwef_ [514]

-----------------------------------------------

            0.00           5.39       5/521            cnfunc_ [60]
            0.00         167.14       155/521          objfun_ [16]
            0.01         389.26       361/521          confun_ [12]
[6] 75.7    0.01         561.79       521              traj_ [6]
            0.28         549.26       1563/1563        phzxm_ [8]
            0.00          10.53       520/520          setic_ [45]
            0.00           1.50       1563/1563        phzxmi_ [119]
            0.00           0.17       30/30            savic_ [188]
            0.00           0.04       1043/1043        dinpt_ [249]
            0.01           0.00       1563/1563        setiv_ [335]
            0.00           0.00       1563/1563        clspfl_ [384]
            0.00           0.00       3/25556          __do_f_out_nv [66]
            0.00           0.00       1042/366616      __s_copy [217]
            0.00           0.00       1/210            pager_ [241]
            0.00           0.00       1/3541           __e_wsfe [153]
            0.00           0.00       1/3541           __s_wsFe_nv [338]
            0.00           0.00       3126/3126        calf_ [460]

-----------------------------------------------

            0.00         557.97       5/5              nlprg_ [5]
[7] 75.2    0.00         557.97       5                npsol_ [7]
            0.00         385.35       5/5              npcore_ [13]
            0.00         172.18       5/5              npchkd_ [15]
            0.00           0.21       5/16             nomout_ [152]
            0.00           0.15       5/5              npdflt_ [194]
            0.00           0.06       5/21             lscore_ [181]
            0.00           0.01       5/7              mchpar_ [278]
            0.00           0.01       160/248          cmqmul_ [312]
            0.00           0.01       5/5              dgeqr_ [345]
            0.00           0.00       5/5              cmchk_ [351]
            0.00           0.00       10/3541          __e_wsfe [153]
            0.00           0.00       5/25556          _do_f_out_nv [66]
            0.00           0.00       335/674          dcopy_ [408]
            0.00           0.00       165/648          dload_ [409]
            0.00           0.00       15/15817893      __pow [30]
            0.00           0.00       10/3541          __s_wsFe_nv [338]
            0.00           0.00       20/62            icopy_ [440]
```

```
                          Called/Total        Parents
Index      %Time        Self Descendents    Called+Self           Name      Index
                          Called Total        Children

             0.00           0.00           5/5                   nploc_ [447]
             0.00           0.00           5/72                  dcond_ [437]
             0.00           0.00           5/357                 dscal_ [417]
             0.00           0.00           5/5                   lscrsh_ [454]
             0.00           0.00           5/19                  f06qhf_ [452]
             0.00           0.00           5/32                  f06qff_ [453]
             0.00           0.00           5/5                   lsbnds_ [506]
             0.00           0.00           5/5                   lssetx_ [507]
             0.00           0.00           5/5                   npcrsh_ [508]

-----------------------------------------------

             0.28         549.26          1563/1563              traj_ [6]
[8]   74.1   0.28         549.26          1563                   phzxm_ [8]
             5.40         537.14          268836/268836          ruk_ [9]
             1.16           2.35          71962/271962           tgoem_ [85]
             0.06           1.50          3126/1082638           motion_ [10]
             0.27           0.88          271962/271962          infxm_ [126]
             0.38           0.00          271962/271962          cycxm_ [161]
             0.11           0.00          271962/271962          dynxm_ [205]
             0.02           0.00          268836/268836          dyns1_ [286]
             0.00           0.00          3126/1081075           deriv_ [122]

-----------------------------------------------

             5.40         537.14          68836/268836           phzxm_ [8]
[9]   73.1   5.40         537.14          268836                 ruk_ [9]
            21.10         514.67          1075344/1082638        motion_ [10]
             1.37           0.00          1075344/1081075        deriv_ [122]

-----------------------------------------------

             0.03           0.75          1563/1082638           motial_ [131]
             0.05           1.25          2605/1082638           tgoem_ [85]
             0.06           1.50          3126/1082638           phzxm_ [8]
            21.10         514.67          1075344/1082638        ruk_ [9]
[10]  72.7  21.24         518.16          1082638                motion_ [10]
            15.48         236.24          1082638/1082638        auxfm_ [14]
            51.67           7.64          218692876/324754409    gentab_ [19]
             6.05          29.52          1082638/1082638        georate_ [26]
            14.38          17.66          1082638/1082638        prop_ [27]
             9.43          22.32          1082638/1082638        aero_ [28]
             1.40          18.88          1082638/1082638        tmotm_ [34]
             6.91          11.64          1082638/1082638        gdgclt_ [36]
             1.21          12.75          1082638/4331073        atmos_ [23]
             2.21           4.06          2165276/2165276        azfpa1_ [53]
             2.38           3.23          1082638/1082638        gamlam_ [58]
             2.65           2.76          1082638/1082638        dgamli_ [59]
             2.94           1.93          4330552/53487770       __cosd [21]
             3.17           1.57          1061798/1061798        guid1_ [65]
```

```
                        Called/Total       Parents
Index      %Time      Self Descendents   Called+Self        Name       Index
                        Called Total      Children

             4.45          0.00        1082638/1084201     mtrxm_ [68]
             2.18          1.50        3247914/52405653    __sind [22]
             1.72          1.80        1082638/1084201     ibmtrx_ [84]
             3.15          0.00        1082638/1082638     dgamla_ [91]
             2.86          0.00        3206234/6454148     mtrxv_ [56]
             1.15          1.05        1082638/1082638     __d_atn2d [101]
             1.92          0.18        2165276/20474779    __atan2 [35]
             1.93          0.14        1082638/1082638     dgamlr_ [105]
             0.58          0.34        3182268/3182268     res180_ [137]
             0.68          0.00        4309712/4309712     vmag_ [150]
             0.37          0.00        274567/274567       monitr_ [162]
             0.09          0.24        1082638/10836279    __sin [87]
             0.32          0.00        4330552/53487770    __d_cosd [76]
             0.28          0.00        3247914/52405653    __d_sind [67]
             0.05          0.22        1082638/10836279    __cos [95]
             0.27          0.00        2123596/16197890    vdot_ [106]
             0.25          0.00        1082638/1082638     wgtm_ [178]
             0.25          0.00        1082638/22694961    __d_sign [61]
             0.12          0.00        2165276/19380158    __d_atn2 [128]
             0.04          0.00        1082638/1082638     calspe_ [254]

-----------------------------------------------

             0.00        165.16        5/16                npchkd_ [15]
             0.00        363.35        11/16               npcore_ [13]
[11] 71.3    0.00        528.51        16                  npfd_ [11]
             0.01        367.76        341/361             confun_ [12]
             0.00        160.74        496/516             objfun_ [16]

-----------------------------------------------

             0.00          5.39        5/361               npchkd_ [15]
             0.00          6.47        6/361               npcore_ [13]
             0.00          9.71        9/361               npsrch_ [38]
             0.01        367.76        341/361             npfd_ [11]
[12] 52.5    0.01        389.33        361                 confun_ [12]
             0.01        389.26        361/521             traj_ [6]
             0.06          0.00        722/1759            chkvec_ [195]
             0.00          0.00        356/872             cmpvec_ [469]

-----------------------------------------------

             0.00        385.35        5/5                 npsol_ [7]
[13] 52.0    0.00        385.35        5                   npcore_ [13]
             0.00        363.35        11/16               npfd_ [11]
             0.00         12.62        6/6                 npsrch_ [38]
             0.00          6.47        6/361               confun_ [12]
             0.00          1.94        6/516               objfun_ [16]
             0.00          0.45        11/16               nomout_ [152]
             0.00          0.25        11/11               npprt_ [177]
```

```
                         Called/Total         Parents
Index      %Time       Self Descendents     Called+Self          Name      Index
                         Called Total         Children

             0.00          0.19          16/16                npiqp_ [185]
             0.00          0.06          5/26                 cmprt_ [172]
             0.00          0.00          20/25556             __do_f_out_nv [66]
             0.00          0.00          5/3541                 __e_wsfe [153]
             0.00          0.00          6/6                  npupdt_ [389]
             0.00          0.00          11/341               .rem [288]
             0.00          0.00          11/248               cmqmul_ [312]
             0.00          0.00          6/452                dgemv_ [277]
             0.00          0.00          5/3541               __s_wsFe_nv [338]
             0.00          0.00          11/11                npmrt_ [438]
             0.00          0.00          62/674               dcopy_ [408]
             0.00          0.00          28/72                dcond_ [437]
             0.00          0.00          28/193               ddot_ [420]
             0.00          0.00          10/366616            __s_copy [217]
             0.00          0.00          6/17                 iload_ [448]
             0.00          0.00          6/32                 f06qff_ [453]
             0.00          0.00          60/404               dnrm2_ [473]
             0.00          0.00          51/253               ddiv_ [476]
             0.00          0.00          11/11                npfeas_ [497]

-----------------------------------------------

            15.48        236.24          1082638/1082638      motion_ [10]
[14] 33.9   15.48        236.24          1082638              auxfm_ [14]
             3.60        140.43          1082638/1082638      conic_ [17]
             3.62         38.25          3247914/4331073      atmos_ [23]
            10.22          5.97          9743742/15817893     __pow [30]
             7.20          0.00          2165276/2165797      mtrxt_ [50]
             2.91          2.00          4330552/52405653     __sind [22]
             2.94          1.93          4330552/53487770     __cosd [21]
             1.14          3.17          1082638/1083159      backor_ [71]
             2.84          0.26          3206234/20474779     __atan2 [35]
             1.93          0.00          2165276/6454148      mtrxv_ [56]
             0.72          1.20          1082638/1082638      irtbr_ [109]
             1.68          0.00          3247914/12124625     __asin [52]
             1.46          0.00          2165276/11909018     vunit_ [48]
             0.50          0.00          2165276/22694961     __d_sign [61]
             0.49          0.00          2165276/10826380     vcross_ [99]
             0.45          0.00          1082638/1082638      momtr_ [156]
             0.37          0.00          4330552/52405653     __d_sind [67]
             0.32          0.00          4330552/53487770     __d_cosd [76]
             0.27          0.00          3247914/37085648     __atan [92]
             0.18          0.00          3206234/19380158     __d_atn2 [128]
             0.14          0.00          1082638/16197890     vdot_ [106]
             0.04          0.02          2605/2605            omtqtn_ [239]

-----------------------------------------------
```

```
                         Called/Total        Parents
Index       %Time      Self Descendents    Called+Self            Name     Index
                         Called Total        Children

            0.00           172.18          5/5                  npsol_ [7]
[15] 23.2   0.00           172.18          5                    npchkd_ [15]
            0.00           165.16          5/16                 npfd_ [11]
            0.00             5.39          5/361                confun_ [12]
            0.00             1.62          5/516                objfun_ [16]
            0.00             0.01          30/25556             __do_f_out_nv [66]
            0.00             0.00          10/3541              __e_wsfe [153]
            0.00             0.00          10/3541              __s_wsFe_nv [338]
            0.00             0.00          10/8233966           .mul [124]
            0.00             0.00          5/17                 iload_ [448]
            0.00             0.00          5/674                dcopy_ [408]
            0.00             0.00          5/648                dload_ [409]
            0.00             0.00          5/19                 f06qhf_ [452]
            0.00             0.00          5/32                 f06qff_ [453]
            0.00             0.00          5/5                  chfd_ [505]

-----------------------------------------------

            0.00             1.62          5/516                npchkd_ [15]
            0.00             1.94          6/516                npcore_ [13]
            0.00             2.92          9/516                npsrch_ [38]
            0.00           160.74          496/516              npfd_ [11]
[16] 22.5   0.00           167.22          516                  objfun_ [16]
            0.00           167.14          155/521              traj_ [6]
            0.08             0.00          1032/1759            chkvec_ [195]
            0.00             0.00          516/872              cmpvec_ [469]

-----------------------------------------------
```

Appendix D

Partial Code from npfd.f

```
c dana 01/03/01 - reversed the loop
c     do 340 j  = 1, n

      print *,'npfd:rev(1):',second(2)
      dana1a=etimedif()
      do 340 j  = n, 1, -1

         xj     = x(j)
         nfound = 0
         if (ncdiff .gt. 0) then
            do 310 i = 1, ncnln
c            --changed cjacu to cjac. it is cjac we wish to fill,
c              and error cjac=rdummy can result if we use cjacu.
c              --d.w.olson   mmc, 6-26-92
               if (cjac(i,j) .eq. rdummy) then
                  needc(i) = 1
                  nfound   = nfound + 1
               else
                  needc(i) = 0
               end if
  310       continue
         end if

         if (nfound .gt. 0  .or.  gradu(j) .eq. rdummy) then
            stepbl = biglow
            stepbu = bigupp
            if (bl(j) .gt. biglow) stepbl = bl(j) - xj
            if (bu(j) .lt. bigupp) stepbu = bu(j) - xj

            if (centrl) then
               if (offset .eq. 1) then
                  delta = dint
               else
                  delta = control(j)
               end if
            else
               if (offset .eq. 1) then
                  delta = feint
               else
                  delta = ford(j)
               end if
            end if

            delta  = delta*(one + abs(xj))
            dorm = max (dorm, delta)
            if (half*(stepbl + stepbu) .lt. zero) delta =  - delta

            x(j) = xj + delta
            if (nfound .gt. 0) then
               call confines( mode, nanny, n, locus,
     $                      needc, x, c1, cjacu, nstate )
               if (mode .lt. 0) go to 999
            end if


c         --changed gradu to grad.  it is grad we wish to fill,
            if (grad(j) .eq. rdummy) then
               call objfun( mode, n, x, objf1, gradu, nstate )
               if (mode .lt. 0) go to 999
            end if

            if (centrl) then
*              ----------------------------------------------------
*              central differences.
*              ----------------------------------------------------
               x(j)  = xj + delta + delta

               if (nfound .gt. 0) then
                  call confun( mode, ncnln, n, ldcju,
     $                         needc, x, c2, cjacu, nstate )
                  if (mode .lt. 0) go to 999

                  do 320 i = 1, ncnln
                     if (needc(i) .eq. 1)
     $                  cjac(i,j) = (four*c1(i) - three*c(i) - c2(i))
     $                                  / (delta + delta)
  320             continue
               end if

               if (gradu(j) .eq. rdummy) then
                  call objfun( mode, n, x, objf2, gradu, nstate )
                  if (mode .lt. 0) go to 999

                  grad(j) = (four*objf1 - three*objf - objf2)
     $                                  / (delta + delta)

               end if
            else
*              ----------------------------------------------------
*              forward differences.
*              ----------------------------------------------------
               if (nfound .gt. 0) then
                  do 330 i = 1, ncnln
                     if (needc(i) .eq. 1)
     $                  cjac(i,j) = (c1(i) -  c(i))/  delta
  330             continue
               end if

               if (gradu(j) .eq. rdummy)
     $            grad(j) = (objf1 - objf) /  delta

            end if
         end if
         x(j) = xj

      print *,'npfd:rev(2):',second(2)
  340 continue
```

```
   dana2a=etimedif()
   danatota=danatota+(dana2a-dana1a)
print *,'npfd:danatota=',danatota
```

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** November 2001	**3. REPORT TYPE AND DATES COVERED** Contractor Report

4. TITLE AND SUBTITLE Parallelization of Program to Optimize Simulated Trajectories (POST3D)	**5. FUNDING NUMBERS** GSA GS-00T-99-AKD-0209 NASA Task Order L-70750D WU 725-10-31-03
6. AUTHOR(S) Dana P. Hammond	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Raytheon Technical Services Company Information Technology and Scientific Services 41 Research Drive, Hampton, VA 23666	**8. PERFORMING ORGANIZATION REPORT NUMBER** RAE001-CR-0701.00
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Langley Research Center Hampton, VA 23681-2199	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER** NASA/CR-2001-211250
11. SUPPLEMENTARY NOTES Langley Technical Monitor: John J. Korte	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified-Unlimited Subject Category: 61 Distribution: Standard Availability: NASA CASI (301) 621-0390	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

This paper describes the parallelization of the Program to Optimize Simulated Trajectories (POST3D). POST3D uses a gradient-based optimization algorithm that reaches an optimum design point by moving from one design point to the next. The gradient calculations required to complete the optimization process, dominate the computational time and have been parallelized using a Single Program Multiple Data (SPMD) on a distributed memory NUMA (non-uniform memory access) architecture. The Origin2000 was used for the tests presented.

14. SUBJECT TERMS Computer Program; Parallelization; Trajectories			**15. NUMBER OF PAGES** 34
			16. PRICE CODE A03
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL