NASA Technical Memorandum 101462

Parallel Processing of a Rotating Shaft Simulation

Dale J. Arpasi
Lewis Research Center
Cleveland, Ohio

February 1989

NASA

PARALLEL PROCESSING OF A ROTATING SHAFT SIMULATION

Dale J. Arpasi
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

SUMMARY

A Fortran program describing the vibration modes of a rotor-bearing system is analyzed for parallelism, and a data-flow statement of the problem is developed. This statement identifies the inherent parallelism in this simulation using a pascal-like structured language. Potential vector operations are also identified.

A critical path through the simulation is identified and used in conjunction with somewhat ficticious processor characteristics to determine the time to calculate the problem on a parallel processing system having those characteristics. A parallel processing overhead time is included as a parameter for proper evaluation of the gain over serial calculation. The serial calculation time is determined for the same ficticious system. An improvement of up to 640 percent is possible depending upon the value of the overhead time.

Based on the analysis, certain conclusions are drawn pertaining to the development needs of parallel processing technology, and to the specification of parallel processing systems to meet specific computational needs.

E-4290

INTRODUCTION

The work described in this report resulted from the cooperative efforts in parallel processing underway in the Internal Fluid Mechanics Division and the Structures Division at the Lewis Research Center. The objectives of these efforts are to establish the requirements of parallel processing to meet the computational needs of these divisions, and to commonly advocate and pursue parallel processing technology based on these requirements.

A number of benchmark simulations are being reviewed to establish the requirements for parallel processing technology development. One of these is a structural dynamics model of a rotor-bearing system. The system is diagramed in figure 1. It consists of a shaft and three disks mounted on two axially preloaded ball bearings. The bearings were mounted in a squeeze-film damper journal containing a centering spring. The calculation approach is described in reference 1. The purpose of the simulation is to identify the shaft vibration at each time step and to display the vibrations in terms of a motion picture. Fixed time steps of 0.12 msec are considered satisfactory to define the system dynamics. In the simulation, an external forcing function providing shaft position, velocity and acceleration is assumed to be available for sampling at each time step.

The Fortran program in appendix A is used as the basis for identifying the parallelism in the simulation. The resulting data-flow statement is then used to estimate the critical path time through the calculations. The calculation time of the critical path is the minimum achievable parallel processing

time and therefore may be used to evaluate the capabilities of specific processors to do the calculation. Parallel processing and serial processing times are determined using operations and execution times of a ficticious but representative parallel processing system. A discussion of parallel processing overhead is provided. Finally, certain conclusions are drawn based upon the results of this effort.

It is intended that the information and discussion presented will provide input for expert system software development (automation of the development of time-optimized parallel models on any given parallel processing systems), and for benchmark evaluations of available and projected parallel processing systems to meet the computational needs of simulations of this type.

PROBLEM STATEMENT

The Fortran calculation of the rotating shaft simulation contains three main programs: initialization, calculation, and motion picture development. These programs operate independently, linked together by data sets. Taking some liberties with the structure of the Fortran program, the following statement of the simulation was formulated:

```
PROGRAM SHAFT
BEGIN
  READ[GEOMETRY/SHAFT_DATA_FILE1];
  INITIALIZE[SHAFT/GEOMETRY];
  READ[START/TERMINAL];
  READ[IC/SHAFT_DATA_FILE2]
  RESET[PARAMS/START,IC];
  READ[ANGIC/ANGLE_INPUT_DEVICE];
  NEWRAP[SOLUTION/SHAFT,IC,START];
  IF CONVERGED THEN
  WHILE NOT STOP DO
  BEGIN
    READ[STOP/TERMINAL];
    READ[ANGIC/ANGLE_INPUT_DEVICE];
    CALCULATE[RESULTS/ANGIC];
    DISPLAY[GRAPHICS_DEVICE/RESULTS]
  END
END;
```

where GEOMETRY, SHAFT, START, IC, PARAMS, SOLUTION, ANGIC, and RESULTS are groups of variables and INITIALIZE, RESET, READ, NEWRAP, CALCULATE, and DISPLAY are computational tasks which form the next lower level of simulation statement.

The INITIALIZE Procedure allows specification of shaft geometry; the RESET procedure develops initial conditions and time parameters; the CALCULATE procedure identifies the dynamics of the shaft as a function of shaft rotation; the NEWRAP procedure solves the shaft model at a time point using a Newton-Raphson method; the DOMOVIE procedure develops the dynamics into a motion picture showing the vibrations of the shaft during acceleration. A simple data-flow analysis of the program indicates that READ [ANGIC], CALCULATE and DOMOVIE can be done in parallel, but they are pipelined, meaning that while

READ gets a new angle (t = 0), CALCULATE works with the one obtained before (at time t = 1), at the same time DOMOVIE displays the results from t = 2.

DATA-FLOW STATEMENT

The data-flow statement of the simulation, derived from the Fortran, is in appendix B. It is a series of statements which must be computed serially because the results generated by one statement are arguments of subsequent statements. These statements are numbered to indicate their computational sequence. Each serial statement, however, may consist of any number of other statements some of which are data-flow-independent of each other and may be computed in parallel, some of which are serial, forming other data-flow paths, and some of which control the path of calculation according to calculation results.

This simulation statement is not unique since various operations may be shifted from one statement to another through the use of dummy variables. Also, statements which may be computed in parallel within a serial statement, may often be moved forward or backward in the calculation sequence as along as data-flow requirements are not violated. In developing this data-flow statement the author attempted to consolidate the calculations into vector operations (defined here as a single calculation sequence operating on lists of arguments to produce a list of results where the lists are of equal length). This was done to aid in the visualization of the calculations as a vector process. Actual vectorization, of course, is very dependent on the computational hardware involved, including its capabilities in handling indirect access of argument vector values, and in the distribution of result vector values to other calculations.

The data-flow statement is written in a pascal-like structure to simplify its analysis and understanding. The statement is written using procedures which conform to the problem statement described above, and using certain other procedures to avoid repetition and to enhance clarity. The use of the FORWARD statement to link procedures is omitted for the sake of clarity. Other major deviations for pascal are:

1. The DO_IN_PARALLEL statement - indicates that the statements between the following BEGIN/END are data-flow independent and may be computed in parallel.

2. Possible vectors are denoted as comments of the form

{V3[50]:A=B+C}.

where "V3" is a vector process identification, "[50]" implies a vector length of 50, and the remainder the vector calculation.

3. Assignment and equivalence are designated using the equal sign.

4. The FOR_ALL statement is used as a shorthand device. The ":=" is used to mean replacement. That is, the integer variable to the left of the symbol is successively replaced by all values appearing to the right of the symbol to form a multitude of statements. For example,

```
FOR_ALL J:=1..2 X(J)=Y(J)
```

is interpreted as `X(1)=Y(1)` and `X(2)=Y(2)`.

5. Results and arguments are specified as such for all procedures. They are delineated as

```
[RESULTS/ARGUMENTS]
```

and immediately follow the procedure's identification.

6. In specifying constants, `n@k` is interpreted to mean `n` sequential values of `k`.

The data-flow statement takes some liberties with the Fortran variables and procedures to conform to the author's subjective concept of clarity, and to reduce the complexity of the Fortran statements to promote parallism. Many of the Fortran loops had to be expanded and many functions had to be merged into procedures for the sake of computational data-flow (the comments in appendix B (the data-flow statement) try to point out where these functions were in the original Fortran code). For this reason, the data-flow statement differs substantially from the serial Fortran statement of appendix A.

THE CRITICAL PATH

The critical path is the longest calculation path through the simulation. (A path is defined to contain no computational parallelism.) The calculation time of the critical path represents the minimum time in which the problem can be calculated, no matter how many processors are used. The critical path is dependent both on the data-flow in the problem and on the processing speed of the parallel processing system on which the problem is to be calculated (the target processors). Therefore, in a parallel processing environment, it is conducive to success to specify the problem using algorithms which maximize the data-flow paths, and to provide parallel processing hardware which minimizes the calculation times of these paths as well as the overhead (data transfer, synchronization, ...) required for parallel calculation of the problem.

An analysis of the critical path through the Newton-Raphson algorithm can be used to provide insight into the parallel processing of the rotor-bearing simulation. This algorithm represents the bulk of the calculation in this simulation and consumes most of the required computing power in each simulated time step.

The data-flow statement in appendix B is macroscopically a serial calculation if the DO-IN-PARALLEL statements are considered to be computational black-box units. The computational path through the calculation is determined by the control statements. These are statements such as IF, WHILE, and REPEAT...UNTIL, which direct the computation according to the values of computed results. If these control statements are assumed to result in the longest path through the problem then the data-flow statement can be used as an approximation of the critical path. It is only an approximation because exact specification of the critical path depends upon processor calculation times. Using

the more rigorous techniques described in reference 2, along with precise processor operation times, a more computation-time-optimized data-flow statement may result.

Appendix C contains a condensed version of the data-flow statement and appendix D a condensed version of the critical path. In these versions the statements are given in an abbreviated form. For example, the following abbreviated statement appears in the INITIALIZE procedure:

2[172]:148@R=B*C:24@R=B+C*D*E*(F-G);

In this statement, "2" indicates that it corresponds to statement 2 of that procedure, and [172] indicates that this statement consists of 172 parallel statements. The remainder of the abbreviation specifies the number and operational form of the parallel statements. That is, 148 are of the form R=B*C and 24 are of the form R=B+C*D*E*(F-G). In this statement, "R" denotes a real number result. In other statements, "A" is used to specify a complex result, "I" and integer result and "L "a boolean result. The letters to the right of the equal sign are used without any data-type connotation to represent arguments.

The calculation time depends on the characteristics of the target processors. The statement must be broken down into the basic machine operations of that processor and execution times for these operations must be known. Then, if 148 calculations of the type, R=C*C, require more time than 24 calculations of R=B+C*D*E(F-G), then the former is the representation of this statement in the critical path (appendix D), and its calculation time is the calculation time of the statement 2 (the rest being done in parallel). In developing critical path times, it is important to include all operations required by the calculation. Data transfer may be a large part of these calculations if more then one processor is used, or if vectors must be constructed to permit vector processor calculation other overheads must be included in parallel processor calculation times. Accurate statement timing is required for allocation of statement execution to a minimum number of processors and for minimizing data transfer delays. These complications point out the need for expert system software to develop parallel processor programs since the drugery involved is likely to be beyond the patience of most humans.

A shorthand notation opposite to a statement is used to specify the machine operations necessary to calculate a statement. In the example above the notation

#R=148(L+M+S)+24(L+2A+3M+S)

would be used to represent the operations involved in a completely serial calculation of statement 2, and

#R=L+M+S+172X

would be used to represent the operations contributed to the critical path by this statement. The operation mneumonics used in the notation are defined in table I. In the notation for the critical path, an overhead tax is imposed on each result calculated. It represents the data transfer and other overhead

associated with parallel processing. The operational equivalent of this tax is represented by X. Therefore, in the example, the parallel execution of the statement would require 1 load, 1 multiply, 1 store and 172 overhead operations (one for each result computed).

The machine operations given in appendix C represent those necessary for serial calculation of the Newton-Raphson algorithm. Table II gives a count of these operations in the main procedure, NEWRAP, and in all of its service procedures. A total count for the complete algorithm is also given. The operations required for real number results are given in part (a) of the table, and those for complex results in part (b) of the table. Results of other datatypes are lumped into part (a). It was assumed that the algorithm requires 20 iterations to converge and that the rotor is defined by 24 segments. On this basis, the algorithm, when executed serially, requires 500 repetitions of the CALC1 procedure, 480 of CALC2, 6240 of EXTER, 520 of JOURNL and 20 repetitions of SOLVE, for each calculation of the NEWRAP main procedure. If convergence is successful then these are a good approximation to the operations required for the CALCULATE procedure at each time step.

A similar operational analysis of the critical path (appendix D) is given in table III. Note the reductions in repetitions required for most service procedures, the reduction in most operations, and the addition of the overhead tax operator, X, to the operational requirements of each procedure.

Table IV provides a comparison of serial execution time to parallel (critical path) execution time. The numbers of operations required are converted to execution time using the operation times given in table I. These operation times are representative of those associated with the Motorola 68020 processor, operating with cache memory and the companion floating point coprocessor. Note that serial calculation of the problem is projected to take about 2.4 sec. Parallel calculation time is given by the equation.

$$\text{time(msec)} = 380.07 + 220233\,Tx$$

where Tx is the execution time associated with the overhead tax operator. Tx is not specified in table I since it related more to the parallel processing system then it is to an individual processor. The plot of this equation is given in figure 2. It shows that a calculation time improvement approaching 6.4 times the serial calculation may be possible (assuming $Tx = 0$) with parallel processing using these M68020 type processors in a configuration which allows calculation in the critical path time without any significant overhead ($Tx = 0$). On the other hand, an overhead of only 9.33 μs causes parallel processing time to exceed serial processing time. It is therefore extremely important to minimize this overhead time in developing parallel processing hardware and software, and to insure that the programming of the parallel processing system does not increase this overhead.

This potential improvement may not warrant the expense associated with parallel processing (time, manpower and materials), or, it may not be sufficient to meet simulation objectives. Two avenues for potential improvement of the situation are available: (1) find a parallel processing system which further reduces critical path calculation time, or, (2) find a different algorithm for the simulation which provides a shorter critical path (more parallelism). Doing the latter would also require identification of a suitable parallel processing system.

CONCLUDING REMARKS

The effort seems to substantiate the following conclusions:

1. Fortran is difficult to interpret, not self-documenting, and certainly not suited to straight-forward data-flow analysis. Additionally, it is not sufficiently structured, nor is it a sufficiently high order language to prohibit the user from adversely affecting the overhead associated with parallel processing.

2. Parallel processing systems should be thoroughly benchmarked for all intended applications. The parallel to serial calculation time ratios should be determined. Then, before selecting a system, any processing-time improvements should be weighed against initial and ongoing costs.

3. Studies of this kind, although valuable in benchmark evaluations of systems, are very tedious and time consuming (about 6 man weeks from Fortran). Automation of these procedures and those associated with the programming and operation of parallel processors could eliminate errors, minimize overhead, and reduce the turn-around time.

More generally, parallel processing is a valuable computational tool whose usefulness may be significantly impaired by (1) unsuitable computational algorithms, (2) error or inefficiency in parallel program generation, (3) too much operational overhead in the system software, and (4) a poor choice of system hardware to meet application needs. The elimination of these pitfalls might be possible using knowledge-based interfaces between users and the parallel processing systems at their disposal. These interfaces could provide a hardware transparency to the users while minimizing code development time and insuring cost effective utilization of existing resources. Without such interfaces, the costs of developing and adapting codes to new parallel machines may significantly impair new technology in the areas of computing systems, code generation, or both.

APPENDIX A: FORTRAN STATEMENT

Here are three programs the first calculates the physical properties M and K. The second program calculates the transient motion using the properties calculated by the first program. The third program produces a motion picture, from the transient motion calculated by the second program. The first program extends from line 100 to 8400. Input is in namelist "Geom" and output is in namelist "shaft." There are no subroutines. The second program extends from line 8500 to 43100. Input is in namelist "Start," "IC," and "Shaft." Output is in namelist "Fin" and a binary file, Fortran 7. "IC" is updated and used as a restart file. The main control program runs from line 8500 to 10300. A utility subroutine runs from line 10400 to 25900. It has several entry points; "ZIC," "ORDER," "STEP," "DELT," "INCR," "OUTPUT," "ANGLE," "RADIUS," "BC," "DEQ," and ROTOR. A logical function "NEWRAP" is a non-linear equation solver. It calls 2 subroutines "Jacob" and "Solve" and entry point "Angle." "Jacob" calculates the linearized coefficients and "Solve" solves the resulting linearized equation set. "Jacob" calls a subroutine "EXTER" which calculates the external force at some point on the rotor. Several entry points into "Util" are called "Radius," "Rotor," "BC," and "DEQ." "Exter" calls "Journl" which is a program to calculate the force on a Journal/Damper bearing.

The last of three programs produces the 3-D movie of shaft motion. Input is namelist "FRAM" and the binary file on Fortran tape 7. A complex function FO is called which interpolates the motion to equal time steps.

```
0000100      REAL DO(24),DI(24),E(24),RHO(24)
0000200      REAL M(24),IT(24),IP(24),L(24),S(24)
0000300      EQUIVALENCE (N,NUM)
0000400      NAMELIST/SHAFT/NUM,M,IT,IP,K,L,S,SMAX
0000500      REAL*8 AN(2,4),AP(2,4)B(2,4),K(2,2,24,3),DUM,DELT,DS,EI
0000600      REAL DBC(2),EBC(2),LBC(2),EIBC(2),P(2,2)
0000700      DATA P/0.,-1.,1.,0./
0000800      DATA PI,G/3.141593,386.4/
0000900      NAMELIST/GEOM/N,L,DO,DI,E,RHO,M,IT,IP,DBC,EBC,LBC
0001000      READ(5,GEOM)
0001100      WRITE(6,GEOM)
0001200      DO 2 J1=1,2
0001300      EIBC(J1)=EBC(J1)*(PI*DBC(J1)**4)/64
0001400      DO 1 J2=1,4
0001500    1 AP(J1,J2)=0
0001600    2 AP(J1,J1)=1
0001700      DS=LBC(1)
0001800      EI=EIBC(1)
0001900      S(1)=L(1)/2
0002000      AP(1,2)=DS
0002100      AP(2,3)=DS/EI
0002200      AP(1,3)=DS*AP(2,3)/2
0002300      AP(2,4)=AP(1,3)
0002400      AP(1,4)=DS*AP(1,3)/3
0002500      N1=N+1
0002600      DO 8 J=1,N1
0002700      DO 3 J1=1,2
```

```
0002800       DO 3 J2=1,4
0002900     3 AN(J1,J2)=AP(J1,J2)
0003000       DS=LBC(2)
0003100       EI=EIBC(2)
0003200       IF(J.GT.N) GO TO 4
0003300       DS=L(J)/2
0003400       IF(J.GT.1) S(J)=S(J-1)+(L(J-1)+L(J))/2
0003500       DEL=DO(J)**2-DI(J)**2
0003600       SUM=DO(J)**2+DI(J)**2
0003700       A=PI*DEL/4
0003800       DM=2DS*A*RHO(J)/G
0003900       DIP=DM*SUM/8
0004000       DIT=DIP/2+DM*DS**2/3
0004100       EI=A*SUM*E(J)/16
0004200       M(J)=M(J)+DM
0004300       IP(J)=IP(J)+DIP
0004400       IT(J)=IT(J)+DIT
0004500     4 AP(1,2)=DS
0004600       AP(2,3)=DS/EI
0004700       AP(1,3)=DS*AP(2,3)/2
0004800       AP(2,4)=AP(1,3)
0004900       AP(1,4)=DS*AP(1,3)/3
0005000       DO 5 J1=1,2
0005100       DO 5 J2-1,2
0005200       B(J1,J2)=0
0005300       B(J1,J2+2)=0
0005400       DO 5 J3=1,2
0005500       B(J1,J2)=B(J1,J2)+AP(J1,J3)*AN(J3,J2)
0005600     5 B(J1,J2+2)=B(J1,J2+2)+AP(J1,J3)*AN(J3,J2+2)+AP''1,J3+2)*AN(J3,J2)
0005700       DELT=B(1,3)*B(2,4)-B(1,4)*B(2,3)
0005800       DUM=B(2,4)/DELT
0005900       B(2,4)=B(1,3)/DELT
0006000       B(1,3)=DUM
0006100       B(2,3)=-B(2,3)/DELT
0006200       B(1,4)=-B(1,4)/DELT
0006300       IF(J.LT.2) GO TO 7
0006400       DO 6 J1=1,2
0006500       DO 6 J2=1,2
0006600       K(J1,J2,J-1,3)=0
0006700       DO 6 J3=1,2
0006800       K(J1,J2,J-1,3)=K(J1,J2,J-1,3)+P(J1,J3)*B(J3,J2+2)
0006900       DO 6 J4=1,2
0007000       K(J1,J2,J-1,2)=K(J1,J2,J-1,2)-P(J1,J3)*B(J3,J4+2)*B(J4,J2)
0007100     7 IF(J.GT.N) GO TO 9
0007200       DO 8 J1=1,2
0007300       DO 8 J2=1,2
0007400       K(J1,J2,J,1)=0
0007500       K(J1,J2,J,2)=0
0007600       DO 8 J3=1,2
0007700       DO 8 J4=1,2
0007800       K(J1,J2,J,2)=K(J1,J2,J,2)-P(J1,J3)*B(J3,J4)*B(J4,J2+2)
0007900       DO 8 J5=1,2
0008000     8 K(J1,J2,J,1)=K(J1,J2,J,1)+P(J1,J3)*B(J3,J4)*B(J4,J5+2)*B(J5,J2)
0008100     9 SMAX=S(N)+L(N)/2
0008200       WRITE(4,SHAFT)
```

```
0008300      STOP
0008400      END
0008500      COMPLEX*16 U(2,24)
0008600      LOGICAL NEWRAP,FINISH
0008700      CALL ZIC(U,N)
0008800      IF(.NOT.NEWRAP(U,E,N)) STOP
0008900      CALL OUTPUT(FINISH)
0009000    4 CALL ORDER(U)
0009100    1 CALL STEP(1.)
0009200      IF(.NOT.NEWRAP(U,E,N)) GO TO 2
0009300      IF(E.LT.1.E-04) GO TO 3
0009400    2 CALL STEP(-1.)
0009500      CALL DELT(.1)
0009600      GO TO 1
0009700    3 CALL INCR(U)
0009800      IF(E.LT.1.E-06) CALL DELT(2.)
0009900      IF(E.GT.1.E-05) CALL DELT(.5)
0010000      CALL OUTPUT(FINISH)
0010100      IF(FINISH) STOP
0010200      GO TO 4
0010300      END
0010400      SUBROUTINE UTIL(U,R,F,A)
0010500      COMPLEX*16 R(2,3,3),U(2,24),Z(2,24,4)
0010600      COMPLEX*16 A(2,2,2,24,3),C(2,24),F(2),RO(2,3),REF(3)
0010700      INTEGER Q,Q1
0010800      COMPLEX RCG(24,2),RIC(24,2,2),ROIC(2,3),RBC(2,2),REFO(2)
0010900      REAL M(24),IT(24),IP(24),L(24),S(24)
0011000      REAL ANGIC(2),(3),ALP(4,4)
0011100      REAL*8 ANG(3),k(2,2,24,3),T,DT,TEMP,H
0011200       DATA ALP/6*0.,5,0.,2.,3.,1.,0.,6.,11.,6.,1./
0011300       LOGICAL FINISH,LOC
0011400       COMMON/JAC/LOC(24)
0011500       NAMELIST/START/TMAX,AMP,OMEGA
0011600       NAMELIST/FIN/RMAX,T
0011700       NAMELIST/IC/TIC,RIC,RCG,ANGIC,ROIC,RBC,LOC
0011800       NAMELIST/SHAFT/NUM,M,IT,IP,K,L,S,SMAX
0011900       ENTRY ZIC(U,N)
0012000       READ(5,IC)
0012100       WRITE(6,IC)
0012200       READ(5,START)
0012300       WRITE(6,START)
0012400       READ(4,SHAFT)
0012500       WRITE(6,SHAFT)
0012600       WRITE(7) NUM,SMAX,(S(J1),J1=1,NUM),(RCG(J1,1),J1=1,NUM)
0012700       RMAX=0
0012800       T=TIC
0012900       N=NUM
0013000       Q=2
0013100       H=1./OMEGA
0013200       DO 3 J2=1,N
0013300       DO 3 J1=1,2
0013400       TEMP=1/AMP
0013500       IF(J1.EQ.2) TEMP=TEMP*L(J2)
0013600       DO 2 J3=1,2
0013700       Z(J1,J2,J3)=RIC(J2,J1,J3)*TEMP
```

```
0013800     2 TEMP=H*TEMP/J3
0013900     3 Z(J1,J2,3)=0
0014000       RETURN
0014100       ENTRY ORDER(U)
0014200       IF(Q.GT.3) RETUEN
0014300       Q=Q+1
0014400       DO 4 J2=1,N
0014500       DO 4 J1=1,2
0014600     4 Z(J1,J2,Q)=U(J1,J2)/(Q-1)
0014700       RETURN
0014800       ENTRY STEP (BET)
0014900       T=T+BET*H
0015000       Q1=Q-1
0015100       DO 5 J3=1,Q1
0015200       TEMP=1
0015300       J4=J3+1
0015400       DO 5 J5=J4,Q
0015500       TEMP=(J5-1)*BET*TEMP/(J5-J3)
0015600       DO 5 J1=1,2
0015700       DO 5 J2=1,N
0015800     5 Z(J1,J2,J3)=Z(J1,J2,J3)+TEMP*Z(J1,J2,J5)
0015900       RETURN
0001600       ENTRY DELT(BET)
0016100       H=BET*H
0016200       TEMP=1
0016300       DO 6 J3=2,Q
0016400       TEMP=TEMP*BET
0016500       DO 6 J1=1,2
0016600       DO 6 J2=1,N
0016700     6 Z(J1,J2,J3)=TEMP*Z(J1,J2,J3)
0016800       RETURN
0016900       ENTRY INCR(U)
0017000       DO 7 J1=1,2
0017100       DO 7 J2=1,N
0017200       DO 7 J3=1,Q
0017300     7 Z(J1,J2,J3)=Z(J1,J2,J3)+ALP(J3,Q)*U(J1,J2)
0017400       RETURN
0017500       ENTRY OUTPUT(FINISH)
0017600       TO=T
0017700       REFO(1)=REF(1)
0017800       REFO(2)=REF(2)
0017900       WRITE(7)TO,(REFO(J3),(RIC(J2,1,J3),J2=1,N),J3=1,2)
0018000       FINISH=.FALSE.
0018100       IF(T.LT.TMAX)RETURN
0018200       FINISH=.TRUE.
0018300       TIC=T
0018400       DO 8 J3=1,3
0018500       ANGIC(J3)=ANG(J3)
0018600       DO 8 J1=1,2
0018700     8 ROIC(J1,J3)=RO(J1,J3)
0018800       WRITE(3,IC)
0018900       WRITE(6,FIN)
0019000       RETURN
0019100       ENTRY ANGLE
0019200       DT=T-TIC
```

```
0019300       ANG(3)=ANGIC(3)
0019400       ANG(2)=ANGIC(2)+ANGIC(3)*DT
0019500       ANG(1)=ANGIC(1)+ANGIC(2)*DT+.5*ANGIC(3)*DT**2
0019600       DO 9 J1=1,2
0019700       RO(J1,3)=ROIC(J1,3)
0019800       RO(J1,2)=ROIC(J1,2)+ROIC(J1,3)*DT
0019900     9 RO(J1,1)=ROIC(J1,1)+ROIC(J1,2)*DT+.5*ROIC(J1,3)DT**2
0020000       REF(1)=CDEXP((0.,1.)*ANG(1))
0020100       REF(2)=(0.,1.)*ANG(2)*REF(1)
0020200       REF(3)=((0.,1.)*ANG(3)-ANG(2)**2)*REF(1)
0020300       DO 10 J2=1,N
0020400       C(1,J2)=2*AMP*ALP(3,Q)*M(J2)/H**2
0020500       C(2,J2)=(AMP*(2*ALP(3,Q)*IT(J2)/H**2-(0.,1.)*IP(J2)*
0020600      1 (ALP(2,Q)*ANG(2)/H+ALP(1,Q)*ANG(3))))/L(J2)**2
0020700       DO 10 J1=1,2
0020800       U(J1,J2)=0
0020900    10 CONTINUE
0021000       RETURN
0021100       ENTRY RADIUS (R,U,JJ1,JJ2,J)
0021200       TEMP=AMP
0021300       IF(JJ1.EQ.2) TEMP=TEMP/L(J)
0021400       DO 11 J3=1,3
0021500       R(JJ1,JJ2,J3)=(Z(JJ1,J,J3)+ALP(J3,Q)*U(JJ1,J))*TEMP
0021600    11 TEMP=J3*TEMP/H
0021700       RIC(J,JJ1,1)=R(JJ1,JJ2,1)
0021800       RIC(J,JJ1,2)=R(JJ1,JJ2,2)
0021900       ABSR=CABS(RIC(J1,1,1))
0022000       IF(ABSR.GT.RMAX) RMAX=ABSR
0022100       RETURN
0022200       ENTRY BC(R,JJO,JJ2)
0022300    13 DO 14 J3=1,3
0022400       DO 14 J1=1,2
0022500    14 R(J1,JJ2,J3)=REF(J3*RBC(JJO,J1)
0022600       RETURN
0022700       ENTRY DEQ(F,R,J)
0022800       F(1)=(-R(1,2,3)-RCG(J,1)*REF(3)-RO(1,3))*M(J)
0022900      1  -K(1,1,J,1)*R(1,1,1)-K(1,1,J,2)*R(1,2,1)-K(1,1,J,3)*R(I,3,1)
0023000      2  -K(1,2,J,1)*R(2,1,1)-K(1,2,J,2)*R(2,2,1)-K(1,2,J,3)*R(2,3,1)
0023100       F(2)=((-R(2,2,3)-RCG(J,2)*REF(3)-RO(2,3))*IT(J)  -
0023200      1  +(0.,1.)*IP(J)*(ANG(2)*(R(2,2,2)+RCG(J,2)*REF(2)+RO(2,2))  -
0023300      2                  +ANG(3)*(R(2,2,1)+RCG(J,2)*REF(1)+RO(2,1)))
0023400      3  -K(2,1,J,1)*R(1,1,1)-K(2,1,J,2)*R(1,2,1)-K(2,1,J,3)*R(1,3,1)  -
0023500      4
-K(2,2,J,1)*R(2,1,1)-K(2,2,J,2)*R(2,2,1)-K(2,2,J,3)*R(2,3,1))/L(J)
0023600       RETURN
0023700       ENTRY ROTOR(A)
0023800       DO 15 J1=1,2
0023900       DO 15 J2=1,2
0024000       DO 15 J3=1,2
0024100       DO 15 J4=1,N
0024200       DO 15 J5=1,3
0024300    15 A(J1,J2,J3,J4,J5)=0
0024400       DO 16 J4=1,N
0024500       DO 16 J1=1,2
0024600       A(J1,1,J1,J4,2)=-C(J1,J4)
```

```
0024700      A(J1,2,J1,J4,2)=-(0.,1.)*C(J1,J4)
0024800      DO 16 J3=1,2
0024900      DO 16 J5=1,3
0025000      JJ=J4+J5-2
0025100      IF(JJ.LT.1.OR.JJ.GT.N) GO TO 16
0025200       TEMP=AMP*ALP(1,Q)
0025300       IF(J3.EQ.2) TEMP=TEMP/L(JJ)
0025400       IF(J1.EQ.2) TEMP=TEMP/L(J4)
0025500       A(J1,1,J3,J4,J5)=A(J1,1,J3,J4,J5)-K(J1,J3,J4,J5)*TEMP
0025600       A(J1,2,J3,J4,J5)=A(J1,2,J3,J4,J5)-(0.,1.)*K(J1,J3,J4,J5)*TEMP
0025700     16 CONTINUE
0025800      RETURN
0025900      END
0026000      LOGICAL RUNCTION NEWRAP(U,E,N)
0026100      COMPLEX*16 A(2,2,2,24,3),F(2,24),DU(2,24),U(2,24)
0026200      CALL ANGLE
0026300      DO 2 ITTER=1,20
0026400      E=0
0026500      CALL JACOB(A,F,U,ITTER,N)
0026600      CALL SOLVE(A,F,DU,N)
0026700      NEWRAP=.TRUE.
0026800      DO 1 J2=1,N
0026900      DO 1 J1-1,2
0027000      U(J1,J2)=U(J1,J2)+DU(J1,J2)
0027100      ABSU=CDABS(U(J1,J2))
0027200      IF(ABSU.GT.E) E=ABSU
0027300      IF(CDABS(DU(J1,J2)).GT.1.E-10)NEWRAP=.FALSE.
0027400     1 CONTINUE
0027500      IF(NEWRAP) RETURN
0027600     2 CONTINUE
0027700      RETURN
0027800      END
0027900      SUBROUTINE JACOB(A,G,U,ITTER,N)
0028000      COMPLEX*16 A(2,2,2,24,3),F(2,24),DU
0028100      COMPLEX*16 U(2,24),R(2,3,3),FO(2),F1(2)
0028200      LOGICAL LOC,FLAG
0028300      COMMON/JAC/LOC(24)
0028400      FLAG=.FALSE.
0028500      IF(MOD(ITTER,4).EQ.1) FLAG=.TRUE.
0028600      IF(FLAG) CALL ROTOR(A)
0028700      CALL BC(R,1,1)
0028800      DO 1 J1=1,2
0028900     1 CALL RADIUS(R,U,J1,2,1)
0029000      DO 9 14=1,N
0029100      IF(J4,LT.N) GO TO 2
0029200      CALL BC(R,2,3)
0029300      GO TO 4
0029400     2 J5=J4+1
0029500      DO 3 J1=1,2
0029600     3 CALL RADIUS (R,U,J1,3,J5)
0029700     4 CALL DEQ(F(1,J4),R,J4)
0029800      IF(.NOT.LOC(J4)) GO TO 10
0029900      CALL EXTER(FO,R,J4)
0030000      F(1,J4)=F(1,J4)+FO(1)
0030100      F(2,24)=F(2,J4)+FO(2)
```

```
0030200       IF(.NOT.FLAG) GO TO 10
0030300       DO 8 J5=1,3
0030400       J=J4+J5-2
0030500       IF(J.LT.1.OR.J.GT.N) GO TO 8
0030600       DO 7 J3=1,2
0030700       ABSDU=1.E-08
0030800       DU=ABSDU
0030900       DO 6 J2=1,2
0031000       U(J3,J)=U(J3,J)+DU
0031100       CALL RADIUS(R,U,J3,J5,J)
0031200       CALL EXTER(F1,R,J4)
0031300       DO 5 J1=1,2
0031400     5 A(J1,J2,J3,J4,J5)=A(J1,J2,J3,J4,J5)+(F1(J1)-F0(J1))/ABSDU
0031500       U(J3,J)=U(J3,J)-DU
0031600     6 DU=(0.,1.)*DU
0031700       CALL RADIUS(R,U,J3,J5,J)
0031800     7 CONTINUE
0031900     8 CONTINUE
0032000    10 DO 9 J1=1,2
0032100       DO 9 J2=1,2
0032200       DO 9 J3=1,3
0032300     9 R(J1,J2,J3)=R(J1,J2+1,J3)
0032400       RETURN
0032500       END
0032600       SUBROUTINE SOLVE(A,F,DU,N)
0032700       REAL*8 A(4,4,24,3),F(4,24),DU(4,24)
0032800       REAL*8 B(4,4,24,3),C(4,24),BP,BR
0032900       DO 1 IB=1,N
0033000       DO 1 I=1,4
0033100       C(I,IB)=F(I,IB)
0033200       DO 1 JB=1,3
0033300       DO 1 J=1,4
0033400     1 B(I,J,IB,JB)=A(I,J,IB,JB)
0033500       DO 11 IB=1,N
0033600       DO 11 IP=1,4
0033700       BP=B(IP,IP,IB,2)
0033800       DO 2 J=IP,4
0033900       B(IP,J,IB,2)=B(IP,J,IB,2)/BP
0034000       DO 3 J=1,4
0034100     3 B(IP,J,IB,3)=B(IP,J,IB,3)/BP
0034200       C(IP,IB)=C(IP,IB)/BP
0034300       IF(IP.EQ.4) GO TO 7
0034400       I1=IP+1
0034500       DO 6 I=I1,4
0034600       BR=B(I,IP,IB,2)
0034700       DO 4 J=IP,4
0034800       B(I,J,IB,2)=B(I,J,IB,2)-BR*B(IP,J,IB,2)
0034900       DO 5 J=1,4
0035000     5 B(I,J,IB,3)=B(I,J,IB,3)-BR*B(IP,J,IB,3)
0035100     6 C(I.IB)=C(I,IB)=BR*C(IP,IB)
0035200     7 IF(IB.EQ.N) GO TO 11
0035300       DO 10 I=1,4
0035400       BR=B(I,IP,IB+1,1)
0035500       DO 8 J=IP,4
0035600     8 B(I,J,IB+1,1)=B(I,J,IB+1,1)-BR*B(IP,J,IB,2)
```

```
0035700       DO 9 J=1,4
0035800     9 B(I,J,IB+1,2)=B(I,J,IB-1,2)-BR*B(IP,J,IB,3)
0035900    10 C(I,IB+1)=C(I,IB+1)-BR*C(IP,IB)
0036000    11 CONTINUE
0036100       DO 15 IBI=1,N
0036200       IB=N+1-IBI
0036300      DO 15 II=1,4
0036400       I=5-II
0036500       DU(I,IB)=-C(I,IB)
0036600       IF(I.EQ.4) GO TO 13
0036700       J1=I+1
0036800       DO 12 J=J1,4
0036900    12 DU(I,IB)=DU(I,IB)-B(I,J,IB,2)*DU(J,IB)
0037000    13 IF(IB.EQ.N) GO TO 15
0037100       DO 14 J=1,4
0037200    14 DU(I,IB)=DU(I,IB)-B(I,J,IB,3)*DU(J,IB+1)
0037300    15 CONTINUE
0037400       RETURN
0037500       END
0037600       SUBROUTINE EXTER(F,R,J)
0037700 C F(2)=TORQUE(J)/L(J)
0037800       COMPLEX*16 F(2),R(2,3,3)
0037900       REAL*8 DEL
0038000       F(1)=0
0038100       F(2)=0
0038200       IF(J.EQ.5.OR.J.EQ.19) GO TO 1
0038300       GO TO 2
0038400     1 CALL JOURNL(F(1),R(1,2,1),R(1,2,2),0.,005.,00263.,TRUE.)
0038500       F(1)=F(1)-5000.*R(1,2,1)
0038600       RETURN
0038700     2 IF(J.EQ.3.OR.J.EQ.12.OR.J.EQ.21) GO TO 3
0038800       RETURN
0038900     3 DEL=CDABS(R(1,2,1))
0039000       IF(DEL.LE..002) RETURN
0039100       DEL=1.-.002/DEL
0039200       F(1)=-1.D+05*(1.,.1)*DEL*R(1,2,1)
0039300       RETURN
0039400       END
0039500       SUBROUTINE JOURNL(F,R,RDOT,OMEGA,CR,FO,CAV)
0039600       IMPLICIT REAL*8(A-H,O-Z)
0039700       COMPLEX*16 F,R,RDOT,EPS,V,N,(2),A(2),B(2),C(2),DN(2),NSQ
0039800       REAL*8 X(2)
0039900       REAL*4 OMEGA,CR,FO
0040000       LOGICAL CAV
0040100       EQUIVALENCE (EPS,X)
0040200       DATA PI/3.141 592 654/
0040300       EPS=R/CR
0040400       V=RDOT/CR-(0.,1.)*OMEGA*EPS
0040500       F=0
0040600       IF(CDABS(V).EQ.0) RETURN
0040700       EPS=CDABS(V)*EPS/V
0040800       H-EPS*DCONJG(EPS)
0040900       IF(H.GT.1.D-12) GO TO 3
0041000       IF(.NOT.CAC) F=PI*FO*V
0041100       IF(CAV) F=-FO*V*(PI/2+2*(EPS+X(1)))
```

```
0041200       RETURN
0041300     3 D=DSQRT(1-H)
0041400       N(1)=EPS*(1-D)/H
0041500       N(2)=EPS*(1+D)/H
0041600       F=8*FO*V/DCONJG(EPS)**3
0041700       DN(1)=N(1)-N(2)
0041800       DN(2)=DN(1)
0041900       DO 2 I=1,2
0042000       NSQ=N(I)**2
0042100       A(I)=NSQ**(NSQ)+1)/DN(I)**3
0042200       B(I)=(2*N(I)*(2*NSQ+1)/DN(I)**2-3*A(I))/DN(I)
0042300       C(I)=(((6*NSQ+1)/DN(I)-3*A(I))/DN(I)-3*B(I))/DN(I)
0042400       IF(CAV) GO TO 1
0042500       F=PI*C(1)*F
0042600       RETURN
0042700     1 A(I)=N(I)*A(I)/(NSQ+1)**2
0042800     2 B(I)=B(I)/(NSQ+1)
0042900       F=(-A(1)-A(2)+B(1)+B(2)-C(1)*DATAN2(-D,X(1)))*F
0043000       RETURN
0043100       END
0043200       REAL VARS(7),DVAR(7),CL(10),DCL(10),LABLE(8)
0043300       REAL X(24),Y(24),S(24),XCG(24),YCG(24),T(2)
0043400       INTEGER IVAR(3)
0043500       LOGICAL CG
0043600       COMPLEX R(2,2,24),REF(2,2),FO,RO,REFO,RCG(24)
0043700       NAMELIST/FRAM/H,RMAX,TMIN,TMAX,NUM,CG,LABLE
0043800       DATA DVAR/7.,2.5,0.,0.,1.,1.,2./
0043900       DATA DCL/-1.,-.6,-.55,-.45,-.4,.4,.45,.55,.6,1./
0044000       READ(5,FRAM)
0044100       READ(7) N,SMAX,(S(J),J=1,N),(RCG(J),J=1,N)
0044200       READ(7) T(1), REF(J3,1),(R(J3,1,J),J=1,N),J3=1,2)
0044300       READ(7) T(2),(REF(J3,2),(R(J3,2,J),J=1,N),J3=1,2)
0044400       CALL GMOVIE(200)
0044500       CALL TITLE(1,32,29,LABLE)
0044600       DO 1 I=1,7
0044700     1 VARS(I)=DVAR(I)
0044800       VARS(5)=RMAX
0044900       DO 2 I=1,10
0045000     2 CL(I)=DCL(I)*RMAX
0045100       IVAR(1)=3
0045200       DO 4 J=1,N
0045300     4 S(J)=RMAX*(2*S(J)/SMAX-1)
0045400       TO=TMIN
0045500     3 VARS(3)=0
0045600       CALL INTENS(40)
0045700       CALL XAXIS(5.,5.,VARS)
0045800       VARS(3)=90
0045900       CALL YAXIS(5.,5.,VARS)
0046000       IVAR(2)=2
0046100       IVAR(3)=0
0046200       DO 5 I=1,5
0046300     5 CALL GPLOT(CL(2*I-1),CL(2*I-1),IVAR)
0046400       CALL INTENS(20)
0046500       DO 10 ITTER=1,NUM
0046600     6 IF(TO.LE.T(2)) GO TO 8
```

```
0046700       T(1)=T(2)
0046800       DO 7 J3=1,2
0046900       REF(J3,1)=REF(J3,2)
0047000       DO 7 J=1,N
0047100     7 R(J3,1,J)=R(J3,2,J)
0047200       READ(7,END=11,ERR=11) T(2),(REF(J3,2),(R(J3,2,J),J=1,N),J3=1,2)
0047300       GO TO 6
0047400     8 REFO=FO(REF,T,TO)
0047500       DO 9 J=1,N
0047600       RO=FO(R(1,1,J),T,TO)-(1.,1.)*S(J)
0047700       X(J)=REAL(RO)
0047800       Y(J)=AIMAG(RO)
0047900       RO=RO+RCG(J)*REFO
0048000       XCG(J)=REAL(RO)
0048100     9 YCG(J)=AIMAG(RO)
0048200       IVAR(2)=N
0048300       IVAR(3)=0
0048400       CALL GPLOT(X,Y,IVAR)
0048500       IVAR(3)=1
0048600       IF(CG) CALL GPLOT(XCG,YCG,IVAR)
0048700    10 TO=TO+H
0048800       CALL DISPLA(1)
0048900       CALL GMOVIE(NUM)
0049000       IF(TO.LT.TMAX) GO TO 3
0049100    11 CALL TERM
0049200       STOP
0049300       END
0049400       COMPLEX FUNCTION FO(F,X,XO)
0049500       COMPLEX F(2,2),A
0049600       REAL X(2),H(2),D(2)
0049700       FO=0
0049800       D(1)=X(1)-X(2)
0049900       H(1)=((XO-X(2))/D(1))**2
0050000       D(2)=X(2)-X(1)
0050100       H(2)=((XO-X(1))/D(2))**2
0050200       DO 1 J=1,2
0050300       A=F(2,J)-2*F(1,J)/D(J)
0050400     1 FO=FO+H(J)*(F(1,J)+A*(XO-X(J)))
0050500       RETURN
0050600       END
```

APPENDIX B: THE DATA-FLOW STATEMENT

```
PROGRAM SHAFT
CONSTANTS(REAL)
  PI=3.141593;
  G=386.4;
  ALP(4,4)=[6@0.,.5,0.,2.,3.,1.,0.,6.,11.,6.,1.];
  K4160=8.0*.00263/.005;
  K411AO=-.00263*PI/(2*.005);
  K411BO=-.00263*2/.005;
VARIABLES(BOOLEAN)
  LOC(24);
  CONVERGED;
  STOP;
  CURRENT
VARIABLES(INTEGER)
  N(24);  {NUMBER OF SEGMENTS}
  Q(2..4);  {ORDER OF INTEGRATION}
VARIABLES(REAL)
  L(24); {LENGTH OF ELEMENT}
  DO(24);{OUTSIDE DIAMETER OF ELEMENT}
  DI(24);{INSIDE DIAMETER OF ELEMENT}
  S(24); {AXIAL DISTANCE OF ELEMENT'S CENTER}
  M(24); {MASS OF ELEMENT}
  IT(24); {ELEMENT'S TRANSVERSE MOMENT OF INERTIA}
  IP(24); {ELEMENT'S POLAR MOMENT OF INERTIA}
  E(24); {YOUNG'S MODULUS OF ELEMENT}
  RHO(24); {DENSITY OF ELEMENT}
  SMAX; {LENGTH OF SHAFT}
  DBC(2); {DIAMETER OF BOUNDRY ELEMENTS}
  LBC(2); {LENGTH OF BOUNDRY ELEMENTS}
  EBC(2); {YOUNG'S MODULUS OF BOUNDRY ELEMENTS}
  K(2,2,24,3); {STIFFNESS MATRIX}
  ANGIC(3); {INITIAL SHAFT ANGLE PROPERTIES}
  DELTA_ANGLE(3); {TIME-VARIANT SHAFT ANGLE DISPLACEMENT}
  TIC; {INITIAL TIME}
  AMP;
  OMEGA; {FREQUENCY}
  T; {ACCUMULATED TIME}
  TMAX; {MAXIMUM OF ACCUMULATED TIME}
  RMAX;
  H; {NON-DIMENSIONAL TIME STEP}
  ALP2; {RESET PARAMETER}
  P204,P205,P206,P215; {NEWRAP PARAMETERS}
  ERR;
VARIBLES(COMPLEX)
  KL(2,2,2,24,3),KLI(2,2,2,24,3);  {ROTOR PARAMETERS}
  U(2,24); {NON-DIMENSIONAL Q+1 DERIVATIVES}
  Z(2,24,4); {NON-DIMENSIONAL <=Q DERIVATIVES}
  R(2,3,3);
  REF(3);
  RO(2,3);
  REFO(2);
  RIC(24,2,2);
  RCG(24,2);
  ROIC(2,3);
```

```
NAME_LISTS
  GEOMETRY:N,L,DO,DI,E,RHO,M,IT,IP,DBC,EBC,LBC;
  SHAFT:ALP2,N,M,IT,IP,K,L,S,SMAX;
  IC:RIC,RCG,ROIC,ANGIC,TIC,RBC,LOC;
  START:TMAX,AMP,OMEGA;
  PARAMS:T,H,Q,RMAX,KL,KLI,Z,P204,P205,P206,P215;
  FORCEFUN:DELTA_ANGLE;
  SOLUTION:CONVERGED,U,ERR,RIC;
  RESULTS:N,SMAX,S,T,REFO,RIC,RMAX,RCG;
```

```
BEGIN {SHAFT}
1 READ[GEOMETRY\SHAFT_DATA_FILE1];
2 DO_IN_PARALLEL
  BEGIN
    DELTA_ANGLE=K0;
    INITIALIZE[SHAFT\GEOMETRY];
    READ[START\TERMINAL];
    READ[IC\SHAFT_DATA FILE2]
  END;
3 RESET[PARAMS\SHAFT,START,IC];
4 NEWRAP[SOLUTION\FORCEFUN,PARAMS,SHAFT];
5 IF CONVERGED THEN
  BEGIN  {5(T)}
  1 DO_IN_PARALLEL
    BEGIN
      STOP=FALSE; CURRENT=FALSE
    END;
  2 WHILE NOT STOP DO
    BEGIN
    1 DO_IN_PARALLEL
      BEGIN
        READ[STOP\TERMINAL];
        READ[FORCEFUN\FORCING FUNCTION INPUT DEVICE];
        {*  THESE RESULTS ARE PIPELINED. THE ACTUAL DISPLAY *}
        {*  TAKES PLACE AT T=T-2.                           *}
        CALCULATE[RESULTS\CURRENT,FORCEFUN,PARAMS,SHAFT];
        DOMOVIE[CURRENT\RESULTS]
      END {5(T).2.1}
    END {5(T).2}
  END {5(T)}
END; {SHAFT}
```

```
PROCEDURE INITIALIZE[SHAFT\GEOMETRY];
CONSTANTS(REAL)
  K2=2;
  KP5=1/2;
  KP25=1/4;
  KP167=1/6;
  KPIO64=PI/64;
  KPIO4G=PI/4*G;
  KPIO32G=PI/32*G;
  KPIO64G=PI/64*G;
  KPIO48G=PI/48*G
VARIABLES(REAL);
  DO2(24); DO4(24; DI2(24); DI4(24); L2(24); LR(24); KE(24);
  KLR4(24); KLR32(24); KLR64(24); T13(24); T14A(24); T14B(24); T24(24);
  TDELT(24); DELT(24); AP(2,4,0..25); B(2,4,24); DBC2(2); DBC4(2); KEB(2)
BEGIN {INITIALIZE}
1 DO_IN_PARALLEL
  FOR_ALL J:=1..N AND I:=1..2
  BEGIN {V1[206]:R=B*C}
    ALP2=K2*ALP(3,3); {USED IN RESET}
    S(1)=L(1)*KP5;
    DBC2(I)=DBC(I)*DBC(I);
    DO2(J)=DO(J)*DO(J);
    DI2(J)=DI(J)*DI(J);
    L2(J)=L(J)*L(J);
    LR(J)=L(J)*RHO(J);
    KEB(I)=EBC(I)*KPIO64;
    KE(J)=E(J)*KPIO64;
    AP(1,2,0)=LBC(1)*KP5;
    AP(1,2,J)=L(J)*KP5;
    AP(1,2,N+1)=LBC(2)*KP5;
    AP(1,3,0)=LBC(1)*KP25;
    AP(1,3,J)=L(J)*KP25;
    AP(1,3,N+1)=LBC(2)*KP25;
    AP(1,4,0)=LBC(1)*KP167;
    AP(1,4,J)=L(J)*KP167;
    AP(1,4,N+1)=LBC(2)*KP167;
    DBC2(I)=DBC(I)*DBC(I)
  END;
```

```
2 DO_IN_PARALLEL
  BEGIN
    FOR_ALL J:=1..N AND I:=1..2
    BEGIN {V2A[148]:R=B*C}
      DBC4(I)=DBC2(I)*DBC2(I);
      DO4(J)=DO2(J)*DO2(J);
      DI4(J)=DI2(J)*DI2(J);
      AP(1,3,0)=AP(1,3,0)*AP(1,2,0);
      AP(1,3,J)=AP(1,3,J)*AP(1,2,J);
      AP(1,3,N+1)=AP(1,3,N+1)*AP(1,2,N+1);
      KLR4(J)=LR(J)*KPIO4G;
      KLR32(J)=LR(J)*KPIO32G;
      KLR64(J)=LR(J)*KPIO64G
    END;
    FOR_ALL J:=1..N  {V2B[24]:R=B+C*D*E*(F-G)}
      IT(J)=IT(J)+L2(J)*LR(J)*KPIO48G*(DO2(J)-DI2(J))
  END; {2}
```

```
INITIALIZE (CONTINUED)
3 DO_IN_PARALLEL
  BEGIN
    FOR_ALL J:=2..N {V3A[23]:R=B+C+D}
      S(J)=S(J-1)+AP(1,2,J)+AP(1,2,J-1);
    FOR_ALL J:=1..N
    BEGIN {V3B[72]:R=B+C(D-E)}
      M(J)=M(J)+KLR4(J)*(DO2(J)-DI2(J));
      IP(J)=IP(J)+KLR32(J)*(DO4(J)-DI4(J));
      IT(J)=IT(J)+KLR64(J)*(DO4(J)-DI4(J))
    END;
    FOR_ALL J:=1..N
    BEGIN {V3C[96]:R=B/(C*(D-E))}
      AP(2,3,J)=AP(1,2,J)/(KE(J)*(DO4(J)-DI4(J)));
      AP(1,3,J)=AP(1,3,J)/(KE(J)*(DO4(J)-DI4(J)));
      AP(2,4,J)=AP(1,3,J)/(KE(J)*(DO4(J)-DI4(J)));
      AP(1,4,J)=AP(1,4,J)/(KE(J)*(DO4(J)-DI4(J)))
    END;
    BEGIN {V3D[8]:R=B/(C*D)}
      AP(2,3,0)=AP(1,2,0)/(KEB(1)*DBC4(1));
      AP(2,3,N+1)=AP(1,2,N+1)/(KEB(2)*DBC4(2));
      AP(1,3,0)=AP(1,3,0)/(KEB(1)*DBC4(1));
      AP(1,3,N+1)=AP(1,3,N+1)/(KEB(2)*DBC4(2));
      AP(2,4,0)=AP(1,3,0)/(KEB(1)*DBC4(1));
      AP(2,4,N+1)=AP(1,3,N+1)/(KEB(2)*DBC4(2));
      AP(1,4,0)=AP(1,4,0)/(KEB(1)*DBC4(1));
      AP(1,4,N+1)=AP(1,4,N+1)/(KEB(2)*DBC4(2))
    END
  END; {3}
4 DO_IN_PARALLEL
  BEGIN
    FOR_ALL J:=1..N+1
    BEGIN {V4A[125]:R=B+C}
      B(1,2,J)=AP(1,2,J-1)+AP(1,2,J);
      B(1,3,J)=AP(1,3,J-1)+AP(1,3,J);
      B(1,4,J)=AP(1,4,J-1)+AP(1,4,J);
      B(2,3,J)=AP(2,3,J-1)+AP(2,3,J);
      B(2,4,J)=AP(2,4,J-1)+AP(2,4,J)
    END;
    FOR_ALL J:=1..N+1
    BEGIN {V4B[100]:R=B*C}
      T13(J)=AP(1,2,J)*AP(2,3,J-1);
      T14A(J)=AP(1,2,J)*AP(2,4,J-1);
      T14B(J)=AP(1,3,J)*AP(1,2,J-1);
      T24(J)=AP(2,3,J)*AP(1,2,J-1)
    END
  END; {4}
```

```
INITIALIZE (CONTINUED)
5 DO_IN_PARALLEL
  BEGIN
    FOR_ALL J:=1..N+1
    BEGIN {V5A[50]:R=B+C}
      B(1,3,J)=B(1,3,J)+T13(J);
      B(2,4,J)=B(2,4,J)+T24(J)
    END;
    FOR_ALL J:=1..N {V5B[24]:R=B+C+D}
      B(1,4,J)=B(1,4,J)+T14A(J)+T14B(J)
  END; {5}
6 DO_IN_PARALLEL
  FOR_ALL J:=1..N+1
  BEGIN {V6[48]:R=B*C}
    TDELT(J)=B(1,4,J)*B(2,3,J);
    DELT(J)=B(1,3,J)*B(2,4,J)
  END; {6}
7 DO_IN_PARALLEL
  FOR_ALL J:=1..N+1  {V7[24]:R=B-C}
    DELT(J)=DELT(J)-TDELT(J);
8 DO_IN_PARALLEL
  FOR_ALL J:=1..N+1
  BEGIN {V8[96]:R=B/C}
    B(1,3,J)=B(1,3,J)/DELT;
    B(2,4,J)=B(2,4,J)/DELT;
    B(2,3,J)=B(2,3,J)/DELT;
    B(1,4,J)=B(1,4,J)/DELT
  END; {8}
9 DO_IN_PARALLEL
  BEGIN
    FOR_ALL J:=2 TO N+1
    BEGIN {V9A[120]:A=B}
      K(1,1,J-1,1)=B(2,3,J-1);
      K(2,1,J-1,3)=B(1,3,J-1);
      K(2,2,J-1,3)=B(1,4,J-1);
      K(1,1,J-1,3)=B(2,3,J);
      K(1,2,J-1,3)=B(2,4,J)
    END;
    FOR_ALL J:=2 TO N+1 {V9B[24]:A=-B-C*D}
      K(1,2,J-1,1)=-B(2,4,J-1)-B(1,2,J-1)*B(2,3,J-1);
    FOR_ALL J:=2 TO N+1 {09C[24]:A=B+C*D}
      K(2,1,J-1,1)=B(1,3,J-1)+B(1,2,J-1)*B(2,3,J-1);
    FOR_ALL J:=2 TO N+1 {V9D[24]:A=B+C+D*(E+F+G)}
      K(2,2,J-1,1)=B(1,4,J-1)+B(1,2,J-1)+
                   B(1,2,J-1)*(B(1,3,J-1)+B(2,3,J-1)+B(2,4,J-1));
    FOR_ALL J:=2 TO N+1 {V9E[24]:A=B+C}
      K(1,1,J-1,2)=B(2,3,J-1)+B(2,3,J);
    FOR_ALL J:=2 TO N+1 {V9F[24]:A=B+C+D*E}
      K(1,2,J-1,2=B(2,4,J-1)+B(2,4,J)+B(2,3,J)*B(1,2,J);
    FOR_ALL J:=2 TO N+1 {V9G[24]:A=-B-C}
      K(2,1,J-1,2)=-B(1,3,J-1)-B(1,3,J);
    FOR_ALL J:=2 TO N+1 {V9H[24]:A=-B-C+D*E}
      K(2,2,J-1,2)=-B(1,4,J-1)-B(1,4,J)+B(1,3,J)*B(1,2,J);
    SMAX=S(N)+L(N)*KP5
  END {9}
END; {INITIALIZE}
```

```
PROCEDURE RESET[PARAMS\SHAFT,START,IC];
CONSTANTS(IMAGINARY)
  KKO=0
CONSTANTS(REAL)
  K2=2; KO=0;
BEGIN {RESET}
1 DO_IN_PARALLEL
  BEGIN
    Q=K2;
    FOR_ALL I2=1..2 AND I3:=1..3
    BEGIN {V1A[27]:R=B}
      T=TIC;
      RMAX=KO;
      KL(I2,2,1,1,I3)=KO;
      KL(I2,2,N,3,I3)=KO;
      KLI(I2,2,1,1,I3)=KO;
      KLI(I2,2,N,3,I3)=KO;
    END;
    FOR_ALL J:=1..N
    BEGIN {V1A[48]:A=B}
      Z(1,J,3)=KKO;
      Z(2,J,3)=KKO
    END;
    FOR_ALL J:=1..N-1
    BEGIN {V1C[76]:R=B*C}
      AMP2=AMP*K2;
      AMAL(2)=AMP*ALP(1,2);
      AMAL(3)=AMP*ALP(1,3);
      AMAL(4)=AMP*ALP(1,4);
      LL(J)=L(J)*L(J+1);
      LL(N)=KO*KO;
      ITA(J)=IT(J)*ALP(3,3);
      ITA(N)=IT(N)*ALP(3,3);
      IPA(J)=IP(J)*ALP(2,3);
      IPA(N)=IP(N)*ALP(2,3);
      A2A=AMP*ALP2
    END
  END; {1}
2 DO_IN_PARALLEL
  BEGIN
    FOR_ALL J:=1..N
    BEGIN {V2A[170]:R=B/C}
      H=K1/OMEGA;
      P215(J)=AMP/L(J);  {USED IN NEWRAP RP1}
      AOL(J)=AMAL(Q)/L(J);
      AOL2(J)=AMAL(Q)/L2(J);
      AOLL(J)=AMAL(Q)/LL(J);
      KIAMP=K1/AMP;
      KLAMP(J)=L(J)/AMP;
      AL2(J)=AMP/L2(J);
      A2L2(J)=AMP2/L2(J)
    END;
```

```
    FOR_ALL J:=1..N
    BEGIN {V2B[48]:A=B/C}
      HRIC1(J)=RIC(J,1,2)/OMEGA;
      HRIC2(J)=RIC(J,2,2)/OMEGA
    END
  END; {2}
```

```
RESET (CONTINUED)
3 DO_IN_PARALLEL   {FORTRAN CODE: CALL ZIC}
  BEGIN
    FOR_ALL J:=1..N
    BEGIN {V3A[72]:R=B*C}
      P204(J)=A2A(J)*M(J);    {USED IN NEWRAP C(1,J)}
      P205(J)=A2L2(J)*ITA(J);{USED IN NEWRAP T2(J)}
      P206(J)=AL2(J)*IPA(J) {USED IN NEWRAP C(2,J)}
    END;
    FOR_ALL J:=1..N
    BEGIN {V3B[96]:A=B*C}
      Z(1,J,1)=KIAMP*RIC(J,1,1);
      Z(2,J,1)=KLAMP(J)*RIC(J,2,1);
      Z(1,J,2)=KIAMP*HRIC1(J);
      Z(2,J,2)=KLAMP(J)*HRIC2(J)
    END;
    FOR_ALL I:=1..3 AND J:=2..N-1
    BEGIN {V3C[852]:R=-B*C,A=IMAG[R]]
      KL(1,1,1,1,I)=-K(1,1,1,1)*AMAL(Q),KLI()=IMAG[KL()];
      KL(1,1,1,2,I)=-K(1,1,1,2)*AMAL(Q),KLI()=IMAG[KL()];
      KL(1,1,1,3,I)=-K(1,1,1,3)*AMAL(Q),KLI()=IMAG[KL()];
      KL(1,2,1,2,I)=-K(1,2,1,2)*AOL(1),KLI()=IMAG[KL()];
      KL(1,2,1,3,I)=-K(1,2,1,3)*AOL(2),KLI()=IMAG[KL()];
      KL(2,1,1,1,I)=-K(2,1,1,1)*AOL(1),KLI()=IMAG[KL()];
      KL(2,1,1,2,I)=-K(2,1,1,2)*AOL(1),KLI()=IMAG[KL()];
      KL(2,1,1,3,I)=-K(2,1,1,3)*AOL(1),KLI()=IMAG[KL()];
      KL(2,2,1,2,I)=-K(2,2,1,2)*AOL2(1),KLI()=IMAG[KL()];
      KL(2,2,1,3,I)=-K(2,2,1,3)*AOLL(1),KLI()=IMAG[KL()];
      KL(1,1,J,1,I)=-K(1,1,J,1)*AMAL(Q),KLI()=IMAG[KL()];
      KL(1,1,J,2,I)=-K(1,1,J,2)*AMAL(Q),KLI()=IMAG[KL()];
      KL(1,1,J,3,I)=-K(1,1,J,3)*AMAL(Q),KLI()=IMAG[KL()];
      KL(1,2,J,1,I)=-K(1,2,J,1)*AOL(J-1),KLI()=IMAG[KL()];
      KL(1,2,J,2,I)=-K(1,2,J,2)*AOL(J),KLI()=IMAG[KL()];
      KL(1,2,J,3,I)=-K(1,2,J,3)*AOL(J+1),KLI()=IMAG[KL()];
      KL(2,1,J,1,I)=-K(2,1,J,1)*AOL(J),KLI()=IMAG[KL()];
      KL(2,1,J,2,I)=-K(2,1,J,2)*AOL(J),KLI()=IMAG[KL()];
      KL(2,1,J,3,I)=-K(2,1,J,3)*AOL(J),KLI()=IMAG[KL()];
      KL(2,2,J,1,I)=-K(2,2,J,1)*AOLL(J-1),KLI()=IMAG[KL()];
      KL(2,2,J,2,I)=-K(2,2,J,2)*AOL2(J),KLI()=IMAG[KL()];
      KL(2,2,J,3,I)=-K(2,2,J,3)*AOLL(J),KLI()=IMAG[KL()];
      KL(1,1,N,1,I)=-K(1,1,N,1)*AMAL(Q),KLI()=IMAG[KL()];
      KL(1,1,N,2,I)=-K(1,1,N,2)*AMAL(Q),KLI()=IMAG[KL()];
      KL(1,1,N,3,I)=-K(1,1,N,3)*AMAL(Q),KLI()=IMAG[KL()];
      KL(1,2,N,1,I)=-K(1,2,N,1)*AOL(N-1),KLI()=IMAG[KL()];
      KL(1,2,N,2,I)=-K(1,2,N,2)*AOL(N),KLI()=IMAG[KL()];
      KL(2,1,N,1,I)=-K(2,1,N,1)*AOL(N),KLI()=IMAG[KL()];
      KL(2,1,N,2,I)=-K(2,1,N,2)*AOL(N),KLI()=IMAG[KL()];
      KL(2,1,N,3,I)=-K(2,1,N,3)*AOL(N),KLI()=IMAG[KL()];
      KL(2,2,N,1,I)=-K(2,2,N,1)*AOLL(N-1),KLI()=IMAG[KL()];
      KL(2,2,N,2,I)=-K(2,2,N,2)*AOL2(N),KLI=IMAG[KL()]
    END
  END {3}
END; {RESET}
```

```
PROCEDURE CALCULATE[RESULTS\CURRENT,FORCEFUN,PARAMS,SHAFT];
BEGIN
1 IF CURRENT THEN {1(T)} WAIT ELSE
  BEGIN {1(E)}
  1 WHILE T<=TMAX DO  {* BOTH T & TMAX ARE READ FROM DATA FILE *}
    BEGIN
    1 IF (CONVERGED AND (Q<4)) THEN                {* FORTRAN CODE:  *}
      DO_IN_PARALLEL                               {* ENTRY ORDER(U) *}
      BEGIN                                        {* CALL INC(U)    *}
        FOR_ALL I:=1..2 AND J:=1..N AND K:=Q+1
          Z(I,J,K)=U(I,J)/Q;
        Q=Q+1
      END; {1(E).1.1(T)}
    2 DO_IN_PARALLEL
      BEGIN
        T=T+H;
        CASE Q OF
          3:FORESTEP3[Z,T\Z,T,H];{ CALL ORDER(U); STEP(1) FOR U=3 }
          4:FORESTEP4[Z,T\Z,T,H];{ CALL ORDER(U); STEP(1) FOR U-4 }
        END
      END; {1(E).1.2}
    3 DO_IN_PARALLEL
      BEGIN
        NEWRAP[SOLUTION\PARAMS,IC,START,SHAFT];
        CASE Q OF
          3:BEGIN
              BACKSTEP3[ZBACK,HBACK,TBACK\Z,H,T];
              HALFSTEP3[ZHALF,HHALF,THALF\Z,H,T];
              DOUBLESTEP3[ZDOUBLE,HDOUBLE,TDOUBLE]
            END;
          4:BEGIN
              BACKSTEP4[ZBACK,HBACK,TBACK\Z,H,T];
              HALFSTEP4[ZHALF,HHALF,THALF\Z,H,T];
              DOUBLESTEP4[ZDOUBLE,HDOUBLE,TDOUBLE]
            END
        END {CASE Q}
      END; {1(E).1.3}
    4 IF NOT CONVERGED OR ERR>E04 THEN
      STEP[Z,H,T\ZBACK,HBACK,TBACK] ELSE
      IF ERR<E06 THEN STEP[Z,H,T\ZDOUBLE,HDOUBLE,TDOUBLE] ELSE
      STEP[Z,H,T\ZHALF,HHALF,THALF]
    END; {1(E).1}
  END;{1(E)}
```

```
CALCULATE (CONTINUED)
2 DO_IN_PARALLEL
  BEGIN
    TMAX=K2*TMAX;
    CURRENT=TRUE;
    FOR ALL I3:=1..3
    BEGIN {V2A[5]:R=B}
      TO=T;
      TIC=T;
      ANGIC(I3)=ANG(I3)
    END;
    FOR ALL I2:=1..2 AND I3:=1..3
    BEGIN {V2B[8]:A=B}
      REFO(I2)=REF(I2);
      ROIC(I2,I3)=RO(I2,I3)
    END
  END; {2}
3 WRITE[SHAFT INPUT FILE2\IC]
END; {CALCULATE}
```

```
PROCEDURE BACKSTEP3[ZA(2,24,4):COMPLEX;HA,TA:REAL\Z(2,24,4):COMPLEX;H,T:REAL];
CONSTANTS(REAL)
  K1=1.; BET1=.1; BET2=.01
BEGIN
1 DO_IN_PARALLEL  {FORTRAN CODE: CALL ORDER(3); STEP(-1) }
  BEGIN
    TA=T-H;
    HA=BET1*H;
```

```
    FOR_ALL I:=1..2 AND J:=1..N {V1A[48]:A=B-C+D}
      ZA(I,J,1)=Z(I,J,1)-Z(I,J,2)+Z(I,J,3);
    FOR ALL I:=1..2 AND J:=1..N {V1B[48]:A=(B-C-D)*E}
      ZA(I,J,2)=(Z(I,J,2)-Z(I,J,3)-Z(I,J,3))*BET1;
    FOR ALL I:=1..2 AND J:=1..N
    BEGIN {V1C[96]:A=B*C}
      ZA(I,J,3)=Z(I,J,3)*BET2;
      ZA(I,J,4)=Z(I,J,4)*K1
    END
  END
END; {BACKSTEP3}
 PROCEDURE BACKSTEP4[ZA(2,24,4):COMPLEX;HA,TA:REAL\Z(2,24,4):COMPLEX;H,T:REAL];
CONSTANTS(REAL)
  K1=1.; BET1=.1; BET2=.01; BET3=.001; K3=3.
VARIABLES(IMAGINARY)
  KZ(1..2,1..24)
BEGIN
1 DO_IN_PARALLEL {FORTRAN CODE: CALL ORDER(4); STEP(-1) }
  BEGIN
    TA=T-H;
    HA=BET1*H;
    FOR_ALL I:=1..2 AND J:=1..N
    BEGIN {V1A[96]:A=B*C}
      KZ(I,J)=K3*Z(I,J,4);
      ZA(I,J,4)=Z(I,J,4)*BET3
    END;
    FOR_ALL I:=1..2 AND J:=1..N
    BEGIN {V1B[96]:A=(B-C-D+E)/F}
      ZA(I,J,1)=(Z(I,J,1)-Z(I,J,2)-Z(I,J,4)+Z(I,J,3))*K1;
      ZA(I,J,2)=(Z(I,J,2)-Z(I,J,3)-Z(I,J,3)+KZ(I,J))*BET1
    END;
    FOR_ALL I:=1..2 AND J:=1..N §V1C[48]:A=(B-C*D)*Eé
      ZA(I,J,3)=(Z(I,J,3)-K3*Z(I,J,4))*BET2
  END
END; {BACKSTEP4}
```

```
PROCEDURE FORESTEP3[Z(2,24,4):COMPLEX\Z(2,24,4):COMPLEX];
BEGIN
1 FOR_ALL I:=1..2 AND J:=1..N  { CALL ORDER(3); STEP(1.0) }
  BEGIN {V1[96]:A=B+C+D}
    Z(I,J,1)=Z(I,J,1)+Z(I,J,2)+Z(I,J,3);
    Z(I,J,2)=Z(I,J,2)+Z(I,J,3)+Z(I,J,3);
  END
END; {FORESTEP3}
 PROCEDURE FORESTEP4[Z(2,24,4):COMPLEX\Z(2,24,4):COMPLEX)];
CONSTANTS(REAL)
  K3=3.
VARIABLES(REAL)
  KZ(1..2,1..24)
BEGIN
1 DO_IN_PARALLEL  { CALL ORDER(4); STEP(1) }
  BEGIN
    FOR_ALL I:=1..2 AND J:=1..N {V1A[48]:A=B*C}
      KZ(I,J)=K3*Z(I,J,4);
    FOR_ALL I:=1..2 AND J:=1..N
    BEGIN {V1B[144]:A=B+C+D+E}
      Z(I,J,1)=Z(I,J,1)+Z(I,J,2)+Z(I,J,3)+Z(I,J,4);
      Z(I,J,2)=Z(I,J,2)+Z(I,J,3)+Z(I,J,3)+KZ(I,J);
      Z(I,J,3)=Z(I,J,3)+Z(I,J,4)+Z(I,J,4)+Z(I,J,4)
    END
  END
END; {FORESTEP4}
PROCEDURE HALFSTEP3[ZA(2,24,4):COMPLEX;HA,TA:REAL\Z(2,24,4):COMPLEX;H,T:REAL];
CONSTANTS(REAL)
  K1=1.; BET1=.5; BET2=.25
BEGIN  { FORTRAN CODE: CALL ORDER(3); STEP(.5) }
1 DO_IN_PARALLEL
  BEGIN
    TA=TA;
    HA=HA*BET1
    FOR ALL I:=1..2 AND J:=1..N
    BEGIN {V1A[144]:A=(B*C+D)*E}
      ZA(I,J,1)=(U(I,J)*ALP(1,3)+Z(I,J,1))*K1;
      ZA(I,J,2)=(U(I,J)*ALP(2,3)+Z(I,J,2))*BET1;
      ZA(I,J,3)=(U(I,J)*ALP(3,3)+Z(I,J,3))*BET2
    END;
    FOR_ALL I:=1..2 AND J:=1..N {V1B[48]:A=B}
      ZA(I,J,4)=Z(I,J,4)
  END
END; {HALFSTEP3}
```

```
PROCEDURE HALFSTEP4[ZA(2,24,4):COMPLEX;HA,TA:REAL\Z(2,24,4):COMPLEX;H,T:REAL];
CONSTANTS(REAL)
  K0=0.; K1=1.; BET1=.5; BET2=.25; BET3=.125
BEGIN { FORTRAN CODE: CALL ORDER(4); STEP(.5) }
1 DO_IN_PARALLEL
  BEGIN
    TA=TA;
    HA=HA*BET1;
    FOR ALL I:=1..2 AND J:=1..N
    BEGIN {V1[192]:A=(B*C+D)*E}
      ZA(I,J,1)=(U(I,J)*ALP(1,4)+Z(I,J,1))*K1;
      ZA(I,J,2)=(U(I,J)*ALP(2,4)+Z(I,J,2))*BET1;
      ZA(I,J,3)=(U(I,J)*ALP(3,4)+Z(I,J,3))*BET2;
      ZA(I,J,4)=(U(I,J)*ALP(4,4)+Z(I,J,4))*BET3
    END
  END
END; {HALFSTEP4}
```

```
PROCEDURE DOUBLESTEP4[ZA(2,24,4):COMPLEX;HA,TA:REAL Z(2,24,4):COMPLEX;H,T:REAL]
CONSTANTS(REAL)
  K1=1.; BET1=2.; BET2=4.
BEGIN { FORTRAN CODE: CALL ORDER(3); STEP(2) }
1 DO_IN_PARALLEL
  BEGIN
    TA=TA;
    HA=HA*BET1;
    FOR ALL I:=1..2 AND J:=1..N
    BEGIN {V1A[144]:A=(B*C+D)*E}
      ZA(I,J,1)=(U(I,J)*ALP(1,3)+Z(I,J,1))*K1;
      ZA(I,J,2)=(U(I,J)*ALP(2,3)+Z(I,J,2))*BET1;
      ZA(I,J,3)=(U(I,J)*ALP(3,3)+Z(I,J,3))*BET2
    END;
    FOR_ALL I:=1..2 AND J:=1..N {V1B[48]:A=B}
      ZA(I,J,4)=Z(I,J,4)
  END
END; {DOUBLESTEP3}
PROCEDURE DOUBLESTEP4[ZA(2,24,4):COMPLEX;HA,TA:REAL\Z(2,24,4):COMPLEX;H,T:REAL]
CONSTANTS(REAL)
  K0=0.; K1=1.; BET1=2.; BET2=4.; BET3=8.
BEGIN {FORTRAN CODE: CALL ORDER(4); STEP(2) }
1 DO_IN_PARALLEL
  BEGIN
    TA=(K0*K0+TA)*K1;
    HA=(K0*K0+HA)*BET1;
    FOR_ALL I:=1..2 AND J:=1..N
    BEGIN {V1[192]:A=(B*C+D)*E}
      ZA(I,J,1)=(U(I,J)*ALP(1,4)+Z(I,J,1))*K1;
      ZA(I,J,2)=(U(I,J)*ALP(2,4)+Z(I,J,2))*BET1;
      ZA(I,J,3)=(U(I,J)*ALP(3,4)+Z(I,J,3))*BET2;
      ZA(I,J,4)=(U(I,J)*ALP(4,4)+Z(I,J,4))*BET3
    END
  END
END; {DOUBLESTEP4}
PROCEDURE STEP[Z(2,24,4):COMPLEX;H,T:REAL\ZA(2,24,4):COMPLEX;HA,TA:REAL];
BEGIN  { FORTRAN CODE: ENTRY STEP(BET) }
1 DO_IN_PARALLEL
  BEGIN
    FOR ALL I:=1..2 AND J:=1..N AND K:=1..Q {V1[48]:A=B}
      Z(I,J,K)=ZA(I,J,K);
    H=HA;
    T=TA
  END
END; {STEP}
```

```
PROCEDURE NEWRAP[SOLUTION FORCEFUN,PARAMS,SHAFT];
CONSTANTS(REAL)
  ABSDUR=1.E-08;
CONSTANTS(COMPLEX)
  ABSDUI=(0,1)*ABSDUR;K20=20;K0=0;KP5=0.5;KE10=1E10;
VARIABLES(BOOLEAN)
  FLAG
VARIABLES(INTEGER)
  ITTER
VARIABLES(REAL)
  DT;DT2;H2;IH1;IH2;RRA1;T1;AA1;AA2;RRA2;ANG2;ANG(3);
  ANG(3);CR;CI;T2(24);T3(24);RP1(24);RP2(24);
  ABSR;ABSU(2,24);ABSUDOT(2,24);CC(4,24);B(4,4,24,3);
VARIABLES(COMPLEX)
  TA(2);TR(2);C(2,24);RO(2,3);TA(24);T4(24);
  A(2,2,2,24,3);R(2,3,3);TB1(24);TB2(24);
  G0;G1;G2;G3;G4;G5;G6;G7;G8;G9;G10;G11;G12;G13;G14;G15;
  C0;C1;C2;C3;C4;C5;
  F(2,24);FC5;F0(2);F1(2);
  RA1;RA1DOT;RB1;RB1DOT;RA2;RA2DOT;RB2;RB2DOT;
BEGIN {NEWRAP}
1 DO_IN_PARALLEL {FORTRAN CODE: CALL JACOB }
  FOR_ALL I:=1..3
  BEGIN
    ANGIC(I)=ANGIC(I)+DELTA ANGLE(I);
    DT=T-TIC
  END; {1}
2 DO_IN_PARALLEL
  BEGIN          {FORTRAN CODE: ENTRY (ANGLE) }
    DT2=KP5*DT*DT;
    TA=ANGIC(2)*DT+ANGIC(1);
    ANG(2)=ANGIC(3)*DT+ANGIC(2);
    ANG(3)=K0*DT+ANGIC(3);
    TR(1)=ROIC(1,2)*DT+ROIC(1,1);
    TR(2)=ROIC(2,2)*DT+ROIC(2,1);
    RO(1,2)=ROIC(1,3)*DT+ROIC(1,2);
    RO(2,2)=ROIC(2,3)*DT+ROIC(2,2);
    RO(1,3)=ROIC(1,3);
    RO(2,3)=ROIC(2,3)
  END; {2}
3 DO_IN_PARALLEL
  BEGIN
    ANG(1)=ANGIC(3)*DT2+TA;
    RO(1,1)=ROIC(1,3)*DT2+TR(1);
    RO(2,1)=ROIC(2,3)*DT2+TR(2);
    ANG2=ANG2*ANG2+K0;
    H2=H*H+K0
  END; {3}
4 DO_IN_PARALLEL
  BEGIN
    IH1=1/H; IH2=1/H2; RRA1=AMP/H;
    CR=COS[ANG(1)]; CI=SIN[ANG(1)]
  END; {4}
```

```
NEWRAP (CONTINUED)
5 DO_IN_PARALLEL
  BEGIN
    REF(1)=CPX[CR,CI];
    REF(2)=CPX[0,ANG(2)];
    REF(3)=CPX[ANG2,ANG(3)]
  END; {5}
6 DO_IN_PARALLEL
  BEGIN
    ITTER=K0;
    REF(2)=REF(2)*REF(1);
    REF(3)=REF(3)*REF(1);
    FOR ALL J:=1..N
      BEGIN {V6A[48]:A=B*C;V6B[76]:R=B*C}
      T1=ANG(3)*ALP(1,Q);
      AA1=ALP(1,Q)*AMP;
      AA2=ALP(2,Q)*RRA1;
      RRA2=AMP2*IH2;
      T2(J)=P205(J,Q)*IH2;
      C(1,J)=P204(J,Q)*IH2;
      C(2,J)=P206(J,Q)*IH2;
      RP1(J)=P215(J)*IH2;
      RP2(J)=P215(J)*IH1
    END;
    FOR_ALL I:=1..2 AND J:=1..N {V6C[48]:A=B}
      U(I,J)=K0;
  END; {6}
7 FOR_ALL J:=1..N {V7[24]:A=B*C+D+E}
    C(2,J)=C(2,J)*ANG(2)+T2(J)+T1;
8 WHILE ITTER<K20 DO
  BEGIN
  1 DO_IN_PARALLEL
    BEGIN
      ITTER=ITTER+1;
      ERR=K0
    END; {8.1}
NEWRAP (CONTINUED)
8.2 DO_IN_PARALLEL
    BEGIN
      FOR_ALL J2:=1..2 AND J3:=1..3 {V2A[6]:A=B*C}
        R(J2,1,J3)=REF(J3)*RBC(1,J2);
      CALC1[\1,2];
      IF MOD[ITTER,4]<>1 THEN FLAG=FALSE ELSE
      DO_IN_PARALLEL
      BEGIN
        FLAG=TRUE;
        FOR_ALL  J:=1..N      {FORTRAN CODE: ENTRY ROTOR(A) }
        BEGIN {V2B[480]:A=B}
          A(1,1,1,J,1)=KL(1,1,J,1,Q-1);
          A(1,1,1,J,3)=KL(1,1,J,3,Q-1);
          A(1,1,2,J,1)=KL(1,2,J,1,Q-1);
          A(1,1,2,J,2)=KL(1,2,J,2,Q-1);
          A(1,1,2,J,3)=KL(1,2,J,3,Q-1);
          A(2,1,1,J,1)=KL(2,1,J,1,Q-1);
```

```
        A(2,1,1,J,2)=KL(2,1,J,2,Q-1);
        A(2,1,1,J,3)=KL(2,1,J,3,Q-1);
        A(2,1,2,J,1)=KL(2,2,J,1,Q-1);
        A(2,1,2,J,3)=KL(2,2,J,3,Q-1);
        A(1,2,1,J,1)=KLI(1,1,J,1,Q-1);
        A(1,2,1,J,3)=KLI(1,1,J,3,Q-1);
        A(1,2,2,J,1)=KLI(1,2,J,1,Q-1);
        A(1,2,2,J,2)=KLI(1,2,J,2,Q-1);
        A(1,2,2,J,3)=KLI(1,2,J,3,Q-1);
        A(2,2,1,J,1)=KLI(2,1,J,1,Q-1);
        A(2,2,1,J,2)=KLI(2,1,J,2,Q-1);
        A(2,2,1,J,3)=KLI(2,1,J,3,Q-1);
        A(2,2,2,J,1)=KLI(2,2,J,1,Q-1);
        A(2,2,2,J,3)=KLI(2,2,J,3,Q-1)
      END;
      FOR_ALL J:=1..N
      BEGIN {V2C[48]:A=B-C}
        A(1,1,1,J,2)=KL(1,1,J,2,Q-1)-C(1,J);
        A(2,1,2,J,2)=KL(2,2,J,2,Q-1)-C(2,J)
      END;
      FOR_ALL J:=1..N
      BEGIN {V2D[48]:A=B-IMAG[C }
        A(1,2,1,J,2)=KLI(1,1,J,2,Q-1)-IMAG[C(1,J)];
        A(2,2,2,J,2)=KLI(2,2,J,2,Q-1)-IMAG[C(2,J)]
      END
    END {8.2(E)}
  END; {8.2}
```

```
NEWRAP (CONTINUED)
8.3 FOR J:=1..N DO
    BEGIN
    1 DO_IN_PARALLEL
      BEGIN
        IF J<N THEN CALC1[\J+1,3];
        ELSE FOR_ALL J1:=1..2 AND J3:=1..3 {V1A(E)[6]:A=B*C}
          R(J1,3,J3)=REF(J3)*RBC(2,J1);
        TA(J)=REF(3)*M(J);
        T3(J)=ANG(3)*IP(J);
        T2(J)=ANG(2)*IP(J);
        T4(J)=REF(3)*IT(J)
      END; {8.3.1}
    2 DO_IN_PARALLEL
      BEGIN
        TB1(J)=REF(1)*T3(J);
        TB2(J)=REF(2)*T2(J)
      END; {8.3.2}
      DO_IN_PARALLEL
    3 BEGIN
        BEGIN {V3A[20]:A=B*C}
          G0=R0(1,3)*M(J);
          G1=R(1,2,3)*M(J);
          G3=R(1,1,1)*K(1,1,J,1);
          G4=R(1,2,1)*K(1,1,J,2);
          G5=R(1,3,1)*K(1,1,J,3);
          G6=R(2,1,1)*K(1,2,J,1);
          G7=R(2,2,1)*K(1,2,J,2);
          F(1,J)=R(2,3,1)*K(1,2,J,3);
          C0=R0(2,2)*T2(J);
          C1=R(2,2,2)*T2(J);
          C3=R0(2,1)*T3(J);
          C4=R(2,2,1)*T3(J);
          G8=R0(2,3)*IT(J);
          G9=R(2,2,3)*IT(J)
          G11=R(1,1,1)*K(2,1,J,1);
          G12=R(1,2,1)*K(2,1,J,2);
          G13=R(1,3,1)*K(2,1,J,3);
          G14=R(2,1,1)*K(2,2,J,1);
          G15=R(2,2,1)*K(2,2,J,2);
          F(2,J)=R(2,3,1)*K(2,2,J,3);
        END;
        G2=RCG(J,1)*TA(J);
        C2=RCG(J,2)*TB2(J);
        C5=RCB(J,2)*TB1(J);
        G10=RCG(J,2)*T4(J)
      END; {8.3.3}
```

```
NEWRAP (CONTINUED)
8.3.4 DO_IN_PARALLEL
      BEGIN
        F(1,J)=-(F(1,J)+G0+G1+G2+G3+G4+G5+G6+G7)
        FC5=KI*(C0+C1+C2+C3+C4+C5)
      END; {8.3.4}
    5 F(2,J)=-(F(2,J)+G8+G9+G10+G11+G12+G13+G14+G15-FC5)/L(J);
    6 IF LOC(J) THEN CALC2;
    7 DO_IN_PARALLEL
      BEGIN
        FOR_ALL I1:=1..2 AND I2:=1..2 AND I3:=1..3 {V7A[12]:A=B}
          R(I1,I2,I3)=R(I1,I2+1,I3);
        FOR ALL I:=1..3
        BEGIN {V7B[26]:A=REAL[B]}
          CC(1,J)=REAL[F(1,J)];
          CC(3,J)=REAL[F(2,J)];
          B(1,1,J,I)=REAL[A(1,1,1,J,I)];
          B(3,1,J,I)=REAL[A(2,1,1,J,I)];
          B(1,2,J,I)=REAL[A(1,2,1,J,I)];
          B(3,2,J,I)=REAL[A(2,2,1,J,I)];
          B(1,3,J,I)=REAL[A(1,1,2,J,I)];
          B(3,3,J,I)=REAL[A(2,1,2,J,I)];
          B(1,4,J,I)=REAL[A(1,2,2,J,I)];
          B(3,4,J,I)=REAL[A(2,2,2,J,I)]
        END;
```

```
      FOR_ALL I:=1..3
      BEGIN {V7C[26]:A=IMAG[B]}
        CC(2,J)=IMAG[F(1,J)];
        CC(4,J)=IMAG[F(2,J)];
        B(2,1,J,I)=IMAG[A(1,1,1,J,I)];
        B(4,1,J,I)=IMAG[A(2,1,1,J,I)];
        B(2,2,J,I)=IMAG[A(1,2,1,J,I)];
        B(4,2,J,I)=IMAG[A(2,2,1,J,I)];
        B(2,3,J,I)=IMAG[A(1,1,2,J,I)];
        B(4,3,J,I)=IMAG[A(2,1,2,J,I)];
        B(2,4,J,I)=IMAG[A(1,2,2,J,I)];
        B(4,4,J,I)=IMAG[A(2,2,2,J,I)]
      END
    END {8.3.7}
  END; {8.3}
4 DO_IN_PARALLEL
  BEGIN
    CONVERGE=TRUE;
    SOLVE[U\U,CC,B,N];
  END; {8.4}
5 FOR_ALL I1:=1..2 AND J:=1..N
  BEGIN {V5[48]:A=CABS[B]}
    ABSU(I1,J)=CABS[U(I1,J)];
    ABSUDOT(I1,J)=CABS[DU(I1,J)]
  END; {8.5}
```

```
NEWRAP (CONTINUED)
8.6 DO_IN_PARALLEL
    BEGIN
      FOR I:=1..2 AND J:=1..N DO IF ABSU(I,J)>ERR THEN ERR=ABSU;
      FOR I:=1..2 AND J:=1..N DO IF ABSUDOT(I,J)>KE10 THEN CONVERGE=FALSE
    END; {8.6}
  7 IF CONVERGE THEN ITTER:=K20
  END {8}
END; {NEWRAP}
^P
PROCEDURE CALC1[ I,M:INTEGER];
BEGIN {CALC1}
1 DO_IN_PARALLEL
  BEGIN {V1[10]:A=B+C*D*E}
    RIC(I,1,1)=Z(1,I,1)+ALP(1,Q)*U(1,I))*AMP;
    RIC(I,1,2)=Z(1,I,2)+ALP(2,Q)*U(1,I))*RRA1;
    RIC(I,2,1)=Z(2,I,1)+ALP(1,Q)*U(2,I))*P215(I);
    RIC(I,2,2)=Z(2,I,2)+ALP(2,Q)*U(2,I))*RP1(I);
    R(1,M,1)=Z(1,I,1)+ALP(1,Q)*U(1,I))*AMP;      {FORTRAN CODE:CALL RADIUS}
    R(1,M,2)=Z(1,I,2)+ALP(2,Q)*U(1,I))*RRA1;
    R(1,M,3)=Z(1,I,3)+ALP(3,Q)*U(1,I))*RRA2;
    R(2,M,1)=Z(2,I,1)+ALP(1,Q)*U(2,I))*P215(I);
    R(2,M,2)=Z(2,I,2)+ALP(2,Q)*U(2,I))*RP1(I);
    R(2,M,3)=Z(2,I,3)+ALP(3,Q)*U(2,I))*RP2(I)
  END;
2 ABSR=CABS(RIC(I,1,1);
3 IF ABSR>RMAX THEN RMAX=ABSR
END; {CALC1}
PROCEDURE CALC2
BEGIN
1 EXTER[FO\J,R(1,2,1),R(1,2,2)];
2 DO_IN_PARALLEL
  BEGIN
    F(1,J)=F(1,J)+FO(1);
    F(2,J)=F(2,J)+FO(2) {NOTE FO(2) ALWAYS ZERO}
  END; {2}
3 IF FLAG THEN
  FOR L:=1..3 DO
  BEGIN
  1 JJ:=J+L-2;
  2 DO_IN_PARALLEL
    BEGIN {V2[8]:A=B+C*(D+E)}
      RA1=Z(1,JJ,1)+AA1*(U(1,JJ)+ABSDUR);
      RA1DOT=Z(1,JJ,2)+AA2*(U(1,JJ)+ABSDUR);
      RB1=Z(1,JJ,1)+AA1*(U(1,JJ)+ABSDUI);
      RB1DOT=Z(1,JJ,2)+AA2*(U(1,JJ)+ABSDUI);
      RA2=Z(1,JJ,1)+AA1*(U(2,JJ)+ABSDUR);
      RA2DOT=Z(1,JJ,2)+AA2*(U(2,JJ)+ABSDUR);
      RB2=Z(1,JJ,1)+AA1*(U(2,JJ)+ABSDUI);
      RB2DOT=Z(1,JJ,2)+AA2*(U(2,JJ)+ABSDUI)
    END; {3.2}
  3 DO_IN_PARALLEL
    BEGIN
      EXTER[FA1\J,RA1,RA1DOT];
```

```
      EXTER[FB1\J,RB1,RB1DOT];
      EXTER[FA2\J,RA2,RA2DOT];
      EXTER[FB2\J,RB2,RB2DOT]
    END; {3.3}
  4 DO_IN_PARALLEL {FORTRAN CODE:ENTRY ROTOR(A)}
    BEGIN {V4[8]:A=B+(C-D)/E}
      A(1,1,1,J,L)=A(1,1,1,J,L)+(FA1(1)-FO(1))/ABSDUR;
      A(2,1,1,J,L)=A(2,1,1,J,L)+(FA1(2)-FO(2))/ABSDUR;
      A(1,2,1,J,L)=A(1,2,1,J,L)+(FB1(1)-FO(1))/ABSDUR;
      A(2,2,1,J,L)=A(2,2,1,J,L)+(FB1(2)-FO(2))/ABSDUR;
      A(1,1,2,J,L)=A(1,1,2,J,L)+(FA2(1)-FO(1))/ABSDUR;
      A(2,1,2,J,L)=A(2,1,2,J,L)+(FA2(2)-FO(2))/ABSDUR;
      A(1,2,2,J,L)=A(1,2,2,J,L)+(FB2(1)-FO(1))/ABSDUR;
      A(2,2,2,J,L)=A(2,2,2,J,L)+(FB2(2)-FO(2))/ABSDUR
    END  {3.4}
  END {3}
END; {CALC2}
```

```
PROCEDURE EXTER[FF(2):COMPLEX\RR,RRDOT:COMPLEX;J:INTEGER];
CONSTANTS(REAL)
  K5000=5000.;
  K002=.002;
  K0=0;
  K1=1.;
  KD05=-1.D+05;
  K005=.005;
  K00263=.00263
VARIABLES(REAL)
  DEL
BEGIN
1 DO_IN PARALLEL
  BEGIN
    FF(1)=0; FF(2)=0
  END; {1}
2 IF (J=5 OR J=19) THEN
  BEGIN {2(T)}
  1 JOURNL[FF(1)\RR,RRDOT,K0,K005,K00263,TRUE];
  2 FF(1)=FF(1)-K5000*RR
  END {2(T)}
  ELSE IF (J=3 OR J=12 OR J=21) THEN
  BEGIN {2(E)(T)}
  1 DEL=CABS[RR];
  2 IF DEL>K002 THEN
    BEGIN {2(E)(T).2(T)}
    1 DEL=K1-K002/DEL;
    2 FF(1)=KD05*RR*CPX[DEL,DEL]
    END {2(E)(T).2(T)}
  END {2}
END;{EXTER}
```

```
PROCEDURE JOURNL[F,R,RDOT:COMPLEX;OM,CR,FZ:REAL4;CAV:BOOLEAN];
CONSTANTS(REAL)
  KDM12=1.D-12;
VARIABLES(REAL)
  H;ABSV;D;DD;
VARIABLES(COMPLEX)
  X;V;EPS;EPSH;CEPS;CEPS3;F;FOA;FOB;N1(2);DN1(2);N2(2);N12(2);
  DN2(2);N34(2);N4(2);N26;NS1(2);DN3(2);NS2(2);AA(2);BB(2);CC(2);AAS;
BEGIN {JOURNL}
1 DO_IN_PARALLEL
  BEGIN
    X=R/CR; V=(RDOT-KI*OM*R)/CR
  END; {1}
2 ABSV=CABS[V];
3 IF ABSV<>0 THEN
  BEGIN {3(T)}
  1 EPS=ABS[V]*X/V;
  2 CEPS=CONJ[EPS];
  3 H=EPS*CEPS;
  4 IF H<=KDM12 THEN
    {4(T)} IF NOT CAV THEN {4(T)(T)} F=-PI*FZ*V  ELSE
    BEGIN {4(T)(E)}
    1 DO_IN_PARALLEL
      BEGIN
        FOA=FZ*PI*V/K2;
        FOB=FZ*V*2;
        F=-(REAL[X]+EPS)
      END; {1}
    2 F=F*FOB-FOA
    END {3(T).4(T)(E)} ELSE
    BEGIN {4(E)}
    1 CEPS3=CEPS*CEPS*CEPS;
    2 DO IN PARALLEL
      BEGIN
        F=8*FZ*V/CEPS3;
        EPSH=EPS/HH;
        D=SQRT[1-H]
      END;{2}
    3 DO_IN_PARALLEL
      BEGIN
        N1(1)=(1-D)*EPSH;
        N1(2)=(1+D)*EPSH
      END; {3}
    4 DO_IN_PARALLEL
      BEGIN
        DN1(1)=N1(1)-N1(2);
        DN1(2)=N1(2)-N1(1);
        N2(1)=N1(1)*N1(1);
        N2(2)=N1(2)*N1(2);
        N12(1)=2*N1(1);
        N12(2)=2*N1(2)
      END; {4}
JOURNL (CONTINUED)
```

```
3.4.5 DO_IN_PARALLEL
      BEGIN
        DN2(1)=DN1(1)*DN1(1);
        DN2(2)=DN1(2)*DN1(2);
        N34(1)=K4*N1(1)*N2(1);
        N34(2)=K4*N1(2)*N2(2);
        N4(1)=N2(1)*N2(1);
        N4(2)=N2(2)*N2(2);
        N26(1)=KM6*N2(1);
        NS1(1)=N2(1)+K1;
        NS1(2)=N2(2)+K1
      END; {5}
    6 DO_IN_PARALLEL
      BEGIN
        DN3(1)=DN1(1)*DN2(1);
        DN3(2)=DN1(2)*DN2(2);
        NS2(1)=NS1(1)*NS1(1);
        NS2(2)=NS1(2)*NS1(2)
      END; {6}
    7 DO_IN_PARALLEL
      BEGIN
        AA(1)=(N4(1)+N2(1))/DN3(1);
        AA(2)=(N4(2)+N2(2))/DN3(2);
        BB(1)=(N34(1)+N12(1))/DN3(1);
        BB(2)=(N34(2)+N12(2))/DN3(2);
        CC(1)=(N26+MK1)/DN3(1)
      END; {7}
    8 DO_IN_PARALLEL
      BEGIN
        BB(1)=BB(1)-K3*AA(1)/DN1(1);
        BB(2)=BB(2)-K3*AA(2)/DN1(2);
        CC(1)=CC(1)+K3*AA(1)/DN2(1)
      END; {3(T).4(E).8}
    9 CC(1)=CC(1)+K3*BB(1)/DN1(1);
   10 IF NOT CAV THEN {10(T) F=-PI*CC(1)*F ELSE
      BEGIN {3(T).4(E).10(E)}
      1 DO_IN_PARALLEL
        BEGIN
          DD=ATAN[-D,REAL[X]];
          AA(1)=AA(1)*N1(1)/NS2(1);
          AA(2)=AA(2)*N1(2)/NS2(2);
          BB(1)=BB(1)/NS1(1);
          BB(2)=BB(2)/NS1(2)
        END; {10(E).1}
      2 AAS=AA(1)+AA(2)-BB(2);
      3 F=F*(CC(1)*DD+BB(1)-AAS)
      END {3(T).4(E).10E}
    END {3(T).4(E)}
  END {3(T)}
END; {JOURNL}
```

```
PROCEDURE SOLVE[U(2,24):COMPLEX\U(2,24):COMPLEX; CC(4,24):REAL;
                B(4,4,24,3):REAL; N(24):INTEGER];
CONSTANTS(REAL)
  MK1=-1.
VARIABLES(REAL)
  MDU(4,24); MBD1(4,4)
BEGIN
1 FOR J:=1..N DO
  BEGIN
    IF J<N THEN
    BEGIN {1(T)}
    1 DO IN PARALLEL
      FOR_ALL I1:=1..4
      BEGIN {V1[9]:A=B/C}
        B(1,I1,J,2)=B(1,I1,J,2)/B(1,1,J,2);
        B(1,I1,J,3)=B(1,I1,J,3)/B(1,1,J,2);
        MDU(1,J)=CC(1,J)/B(1,1,J,2)
      END; {1(T).1}
    2 DO_IN_PARALLEL {FORTRAN CODE: CALL BC}
      FOR_ALL K1:=1..4 AND K2:=2..4 AND I1:=1..4
      BEGIN {V2[63]:A=B-C*D}
        B(K2,I1,J,2)=B(K2,I1,J,2)-B(1,I1,J,2)*B(K2,1,J,2);
        B(K2,I1,J,3)=B(K2,I1,J,3)-B(1,I1,J,3)*B(K2,1,J,2);
        B(K1,I1,J+1,1)=B(K1,I1,J+1,1)-B(1,I1,J,2)*B(K1,1,J+1,1);
        B(K1,I1,J+1,2)=B(K1,I1,J+1,2)-B(1,I1,J,3)*B(K1,1,J+1,1);
        CC(K2,J)=CC(K2,J)-MDU(1,J)*B(K2,1,J,2);
        CC(K1,J+1)=CC(K1,J+1)-MDU(1,J)*B(K1,1,J+1,1)
      END; {1(T).2}
    3 DO_IN_PARALLEL
      FOR_ALL I2:=2..4 AND I1:=1..4
      BEGIN {V3[8]:A=B/C}
        B(2,I2,J,2)=B(2,I2,J,2)/B(2,2,J,2);
        B(2,I1,J,3)=B(2,I1,J,3)/B(2,2,J,2);
        MDU(2,J)=CC(2,J)/B(2,2,J,2)
      END; {1(T).3}
    4 DO_IN_PARALLEL
      FOR_ALL K1:=1..4 AND K3:=3..4 AND I2:=2..4 AND I1:=1..4
      BEGIN {V4[48]:A=B-C*D}
        B(K3,I2,J,2)=B(K3,I2,J,2)-B(2,I2,J,2)*B(K3,2,J,2);
        B(K3,I1,J,3)=B(K3,I1,J,3)-B(2,I1,J,3)*B(K3,2,J,2);
        B(K1,I2,J+1,1)=B(K1,I2,J+1,1)-B(2,I2,J,2)*B(K1,2,J+1,1);
        B(K1,I1,J+1,2)=B(K1,I1,J+1,2)-B(2,I1,J,3)*B(K1,2,J+1,1);
        CC(K3,J)=CC(K3,J)-MDU(2,J)*B(K3,2,J,2);
        CC(K1,J+1)=CC(K1,J+1)-MDU(2,J)*B(K1,2,J+1,1)
      END; {1(T).4}
    5 DO_IN_PARALLEL
      FOR_ALL I3:=3..4 AND I1:=1..4
      BEGIN {V5[7]:A=B/C}
        B(3,I3,J,2)=B(3,I3,J,2)/B(3,3,J,2);
        B(3,I1,J,3)=B(3,I1,J,3)/B(3,3,J,2);
        MDU(3,J)=CC(3,J)/B(3,3,J,2)
      END; {1(T).5}
```

```
SOLVE (CONTINUED)
1T. 6 DO_IN_PARALLEL
      FOR K1:=1..4 AND I3:=3..4 AND I1:=1..4
      BEGIN {V6[17]:A=B-C*D}
        B(4,I3,J,2)=B(4,I3,J,2)-B(3,I3,J,2)*B(4,3,J,2);
        B(4,I1,J,3)=B(4,I1,J,3)-B(3,I1,J,3)*B(4,3,J,2);
        B(K1,I3,J+1,1)=B(K1,I3,J+1,1)-B(3,I2,J,2)*B(K1,3,J+1,1);
        B(K1,I1,J+1,2)=B(K1,I1,J+1,2)-B(3,I1,J,3)*B(K1,3,J+1,1);
        CC(4,J)=CC(4,J)-MDU(3,J)*B(4,3,J,2);
        CC(K1,J+1)=CC(K1,J+1)-MDU(3,J)*B(K1,3,J+1,1)
      END; {1(T).6}
    7 DO_IN_PARALLEL
      FOR_ALL I1:=1..4
      BEGIN {V7[6]:A=B/D}
        B(4,4,J,2)=B(4,4,J,2)/B(4,4,J,2);
        B(4,I1,J,3)=B(4,I1,J,3)/B(4,4,J,2);
        MDU(4,J)=CC(4,J)/B(4,4,J,2)
      END; {1(T).7}
    8 DO_IN_PARALLEL
      FOR_ALL K1:=1..4 AND I1:=1..4
      BEGIN {V8[24]:A=B-C*D}
        B(K1,4,J+1,1)=B(K1,4,J+1,1)-B(4,4,J,2)*B(K1,4,J+1,1);
        B(K1,I1,J+1,2)=B(K1,I1,J+1,2)-B(4,I1,J,3)*B(K1,4,J+1,1);
        CC(K1,J+1)=CC(K1,J+1)-MDU(4,J)*B(K1,4,J+1,1)
      END {1(T).8}
    END {1(T)} ELSE
    BEGIN {1(E)}
    1 DO_IN_PARALLEL
      FOR_ALL I1:=1..4
      BEGIN {V1[9]:A=B/C}
        B(1,I1,J,2)=B(1,I1,J,2)/B(1,1,J,2);
        B(1,I1,J,3)=B(1,I1,J,3)/B(1,1,J,2);
        MDU(1,J)=CC(1,J)/B(1,1,J,2)
      END; {1(E).1}
    2 DO_IN_PARALLEL
      FOR K2:=2..4 AND I1:=1..4
      BEGIN  {V2[18]:A=B-C*D}
        B(K2,I1,J,2)=B(K2,I1,J,2)-B(1,I1,J,2)*B(K2,1,J,2);
        B(K2,I1,J,3)=B(K2,I1,J,3)-B(1,I1,J,3)*B(K2,1,J,2);
        CC(K2,J)=CC(K2,J)-MDU(1,J)*B(K2,1,J,2)
      END; {1(E).2}
    3 DO_IN_PARALLEL
      FOR_ALL I2=2..4 AND I1:=1..4
      BEGIN {V3[7]:A=B/C}
        B(2,I2,J,2)=B(2,I2,J,2)/B(2,2,J,2);
        B(2,I1,J,3)=B(2,I1,J,3)/B(2,2,J,2);
        MDU(2,J)=CC(2,J)/B(2,2,J,2)
      END; {1(E).3}
    4 DO_IN_PARALLEL
      FOR_ALL K3=3..4 AND I2=2..4 AND I1:=1..4
      BEGIN {V4[16]:A=B-C*D}
        B(K3,I2,J,2)=B(K3,I2,J,2)-B(2,I2,J,2)*B(K3,2,J,2);
        B(K3,I1,J,3)=B(K3,I1,J,3)-B(2,I1,J,3)*B(K3,2,J,2);
        CC(K3,J)=CC(K3,J)-MDU(2,J)*B(K3,2,J,2)
      END; {1(E).4}
```

```
SOLVE (CONTINUED)
1E. 5 DO_IN_PARALLEL
      FOR_ALL I3=3..4 AND I1:=1..4
      BEGIN {V5[7]:A=B/C}
        B(3,I3,J,2)=B(3,I3,J,2)/B(3,3,J,2);
        B(3,I1,J,3)=B(3,I1,J,3)/B(3,3,J,2);
        MDU(3,J)=CC(3,J)/B(3,3,J,2)
      END; {1(E).5}
    6 DO_IN_PARALLEL
      FOR_ALL I3=3..4 AND I1:=1..4
      BEGIN {V6[7]:A=B-C*D}
        B(4,I3,J,2)=B(4,I3,J,2)-B(3,I3,J,2)*B(4,3,J,2);
        B(4,I1,J,3)=B(4,I1,J,3)-B(3,I1,J,3)*B(4,3,J,2);
        CC(4,J)=CC(4,J)-MDU(3,J)*B(4,3,J,2)
      END; {1(E).6}
    7 DO_IN_PARALLEL
      FOR_ALL I1:=1..4
      BEGIN {V7[6]:A=B/C}
        B(4,4,J,2)=B(4,4,J,2)/B(4,4,J,2);
        B(4,I1,J,3)=B(4,I1,J,3)/B(4,4,J,2);
        MDU(4,J)=CC(4,J)/B(4,4,J,2)
      END {1(E).7}
    END {1(E)}
  END; {1{  {FOR J:=1..N}
2 FOR J:=N..1 DO
  BEGIN
    IF J<N THEN
    BEGIN {2(T)}
    1 DO_IN_PARALLEL
    1 FOR_ALL K1:=1..4 AND I1:=1..4
      BEGIN {V1[16]:A=B*C}
        MBD1(K1,I1)=MDU(K1,J+1)*B(I1,K1,J,3);
      END; {2(T).1}
    2 DO_IN_PARALLEL
      BEGIN
        MDU(4,J)=MDU(4,J)-MBD1(4,1)-MBD1(4,2)-MBD1(4,3)-MBD1(4,4);
        MDU(3,J)=MDU(3,J)-MBD1(3,1)-MBD1(3,2)-MBD1(3,3)-MBD1(3,4);
        MDU(2,J)=MDU(2,J)-MBD1(2,1)-MBD1(2,2)-MBD1(2,3)-MBD1(2,4);
        MDU(1,J)=MDU(1,J)-MBD1(1,1)-MBD1(1,2)-MBD1(1,3)-MBD1(1,4);
      END; {2(T).2}
    3 DO_IN_PARALLEL
      BEGIN
        MDU(3,J)=MDU(3,J)-B(3,4,J,2)*MDU(4,J);
        MDU(2,J)=MDU(2,J)-B(2,4,J,2)*MDU(4,J);
        MDU(1,J)=MDU(1,J)-B(1,4,J,2)*MDU(4,J)
      END; {2(T).3}
```

```
SOLVE (CONTINUED)
2T. 4 DO_IN_PARALLEL
      BEGIN
        MDU(2,J)=MDU(2,J)-B(2,3,J,2)*MDU(3,J);
        MDU(1,J)=MDU(1,J)-B(1,3,J,2)*MDU(3,J)
      END; {2(T).4}
    5 MDU(1,J)=MDU(1,J)-B(1,2,J,2)*MDU(2,J);
    6 DO_IN_PARALLEL
      BEGIN
        U(1,J)=U(1,J)-CPX[MDU(1,J),MDU(2,J)];
        U(2,J)=U(2,J)-CPX[MDU(3,J),MDU(4,J)]
      END {2(T).6}
    END {2(T)} ELSE
    BEGIN {2(E){ }J:=N}
    1 DO_IN_PARALLEL
      BEGIN
        MDU(3,J)=MDU(3,J)-B(3,4,J,2)*MDU(4,J);
        MDU(2,J)=MDU(2,J)-B(2,4,J,2)*MDU(4,J);
        MDU(1,J)=MDU(1,J)-B(1,4,J,2)*MDU(4,J)
      END; {2(E).1}
    2 DO IN PARALLEL
      BEGIN
        MDU(2,J)=MDU(2,J)-B(2,3,J,2)*MDU(3,J);
        MDU(1,J)=MDU(1,J)-B(1,3,J,2)*MDU(3,J)
      END; {2(E).2}
    3 MDU(1,J)=MDU(1,J)-B(1,2,J,2)*MDU(2,J);
    4 DO_IN_PARALLEL
      BEGIN
        U(1,J)=U(1,J)-CPX[MDU(1,J),MDU(2,J)];
        U(2,J)=U(2,J)-CPX[MDU(3,J),MDU(4,J)]
      END {2(E).4}
    END {2}{E}
  END {2} }J:=N..1}
END; {SOLVE}
```

```
PROCEDURE DOMOVIE[CURRENT\RESULTS];
{THE TRANSFER OF DATA FROM CALCULATE TO DOMOVIE NEEDS MORE DEFINITION.
 ALSO ,THE READ(5,FRAM) STATEMENT MUST BE ADDRESSED}
CONSTANTS(REAL)
  DVARS(7)=7.,2.5,0.,0.,1.,1.,2.;
  DCL(10)=-1.,-.6,-.55,-.45,-.4,-.4,.45,.55,.6,1.
VARIABLES(BOOLEAN)
  CG
VARIABLES(COMPLEX)
  R(2,2,24),REF(2,2),FO,RO,REFO,RCG(24)
VARIABLES(REAL)
  X(24),Y(24),S(24),XCG(24),YCG(24),T(2),TO;
  XVARS(7),YVARS(7),CL(10);LABLE(8)
VARIABLES(INTEGER)
  ITTER,IVARS(3)
BEGIN
1 IF CURRENT THEN
  BEGIN {1(T)}
  1 DO_IN_PARALLEL
    BEGIN
      TRANSFER[T(1),REF(J3,1),R(J3,1,J)\TO,REFO(J3),RIC(J,1,J3)];
      TRANSFER[T(2),REF(J3,2),R(J3,2,J)\TO,REFO(J3),RIC(J,1,J3)];
      CURRENT=FALSE;
      GMOVIE(200);
      FOR I:=1..7 DO VECTOR
      BEGIN
        XVARS(I)=DVAR(I);
        YVARS(I)=DVAR(I);
        IVAR(1)=3;
        IVAR(2)=2;
        IVAR(3)=0
      END
    END;
  2 DO_IN_PARALLEL
    BEGIN
      TITLE(1,32,29,LABEL);
      XVARS(3)=K0;
      XVARS(5)=RMAX;
      YVARS(3)=K90;
      YVARS(5)=RMAX;
      TO=TMIN;
      FOR I:=1..10 DO VECTOR
      BEGIN
        CL(I)=DCL(I)*RMAX;
        SS(J)=RMAX*S(J)
      END
    END;
  3 WHILE TO<TMAX DO
    BEGIN
    1 INTENS(K40);  {FORTRAN CODE:CALL INTENS}
    2 XAXIS(K5,K5,XVARS);
    3 YAXIS(K5,K5,YVARS);
    4 FOR I:=1..5 BY 2 DO
        GPLOT(CL(I),CL(I),IVAR);
    5 INTENS(K20);
    6 FOR ITTER:=1..NUM DO
```

```
      BEGIN
      1 WHILE TO>T(2) DO_IN_PARALLEL
        BEGIN
          T(1)=T(2);
          FOR ALL J3=1..2 AND J:=1..N
          BEGIN {V1A[50]:A=B}
            REF(J3,1)=REF(J3,2);
            R(J3,1,J)=R(J3,2,J)
          END;
          FOR_ALL J3=1..2 AND J:=1..N {V1B[48]:TRANSFER}
            TRANSFER[T(2),REF(J3,2),R(J3,2,J)\TO,REFO(J3),RIC(J,1,J3)]
        END;
      2 DO_IN_PARALLEL
        BEGIN
          REFO=FXX[REF,T,TO];
          FOR ALL J:=1..N {V2[24]:A=FXX}
            RO(J)=FXX[R(1,1,J,T,TO)-CPX(SS(J),SS(J)]
        END;
      3 DO_IN_PARALLEL
        BEGIN
          FOR_ALL J:=1..N {V3A[24]:R=REAL[B]}
            X(J)=REAL[RO(J)];
          FOR_ALL J:=1..N {V3B[24]:R=IMAG[B]}
            Y(J)=IMAG[RO(J)];
          FOR_ALL J:=1..N {V3C[24]:R=REAL[B+C*D]}
            XCG(J)=REAL[RO(J)+RCG(J)*REFO];
          FOR_ALL J:=1..N {V3D[24]:R=IMAG[B+C*D]}
            YCG(J)=IMAG[RO(J)+RCG(J)*REFO]
          IVAR(2)=N;
          IVAR(3)=0
        END;
      4 DO_IN_PARALLEL
        BEGIN
          GPLOT[X,Y,IVAR]; {FORTRAN CODE:CALL GPLOT}
          TO=TO+H;
          IVAR(3)=1
        END;
      5 IF CG THEN GPLOT[XCG,YCG,IVAR]
      6 DISPLAY(K1);  {CALL DISPLA}
      7 GMOVIE(NUM)   {CALL GMOVIE}
      END
    END
  END
END; {DOMOVIE}
```

APPENDIX C: CONDENSED DATA-FLOW STATEMENT

```
PROGRAM SHAFT
BEGIN
  1:READ
  2[4]:R=B:INITIALIZE:2@READ;
  3:RESET;
  4:NEWRAP;
  5:IF CONVERGED THEN
    BEGIN
      1[2]:2@L=B;
      2:WHILE NOT STOP DO
           1[4]:2@READ:CALCULATE:DOMOVIE
    END
END;
PROCEDURE INITIALIZE
BEGIN
  1[206]:R=B*C;
  2[172]:148@R=B*C:24@R=B+C*D*E*(F-G);
  3[199]:23@R=B+C+D:72@R=B+C(D-E):96@R=B/(C*(D-E)):8@R=B/(C*D);
  4[216]:120@R=B+C:96@R=B*C;
  5[72]:48@R=B+C:24@R=B+C+D;
  6[48]:R=B*C;
  7[24]:R=B-C;
  8[96]:R=B/C;
  9[289]:120@R=B:24@R=-B-C*D:24@R=B+C*D:24@R=B+C+D*(E+F+G):
        :24@R=B+C:24@R=B+C+D*E:24@R=-B-C:24@R=-B-C+D*E
END; {INITIALIZE}
PROCEDURE RESET
BEGIN
  1[152]:I=B:27@R=B:48@A=B:76@R=B*C;
  2[218]:170@R=B/C:48@A=B/C;
  3[1020]:72@R=B*C:96@A=B*C:852@(R=-B*C,A=IMAG[R]);
END; {RESET}
PROCEDURE CALCULATE
BEGIN
  1[1]:IF CURRENT THEN WAIT ELSE
       BEGIN {1E}
         WHILE T<=TMAX DO
         BEGIN
           1:IF CONVERGED AND Q<4 THEN S1(E)[49]:I=B+C:48@A=B/C;
           2[2]:R=B+C:IF Q=3 THEN FORESTEP3 ELSE FORESTEP4;
           3[2]:NEWRAP:IF Q=3 THEN 1[3]:BACKSTEP3:HALFSTEP3:DOUBLESTEP3
                      ELSE 1[3]:BACKSTEP4:HALFSTEP4:DOUBLESTEP4;
           4:IF NOT CONVERGED OR ERR>E04 THEN STEP{BACK} ELSE
                IF ERR<E06 THEN  STEP§DOUBLEé ELSE STEP{HALF};
        END;{1(E)}
  2[15]:L=B:3@R=B:2@A=B;
  3:WRITE
END; {CALCULATE}
```

```
PROCEDURE BACKSTEP3
BEGIN
  1[194]:R=B-C:R=B*C:48@A=B-C+D:48@A=(B-C-D)*E:96@A=B*C
END;
PROCEDURE BACKSTEP4
BEGIN
  1[242]:R=B-C:R=B*C:48@A=(B-C*D)*E:96@A=(B-C-D+E)*F:96@A=B*C
END;
PROCEDURE FORESTEP3
BEGIN
  1[96]:A=B+C+D
END;
PROCEDURE FORESTEP4
BEGIN
  1[192]:48@A=B*C:144@A=B+C+D+E
END;
PROCEDURE HALFSTEP3
BEGIN
  1[194]:2@R=B*C:48@A=B*C;144@A=(B*C+D)/E
END;
PROCEDURE HALFSTEP4
BEGIN
  1[194]:2@R=B*C:192@A=(B*C+D)/E
END;
PROCEDURE DOUBLESTEP3
BEGIN
  1[194]:2@R=B*C:48@A=B*C:144@A=(B*C+D)*E
END;
PROCEDURE DOUBLESTEP4
BEGIN
  1[194]:2@R=B*C:192@A=(B*C+D)*E
END;
PROCEDURE STEP
BEGIN {STEP}
  1[194]:2@R=B:192@A=B
END {STEP}
```

```
PROCEDURE NEWRAP  #R=28991L+28991S+5380T+28A+1040M+23D+2COS+960CABS+960IMAG
                     +20#R[SOLVE]+500#R[CALC1]+480#R[CALC2]
                  #A=32891L+32890S+14935A+15079M+480D+965CPX+20#A[SOLVE]
                     +500#A[CALC1]+480#A[CALC2]
BEGIN {NEWRAP}
  1[4]:3@R=B+C:R=B-C;                              #R=4*(L+A+S)
  2[10]:2@A=B:5@A=B*C+D:2@R=B*C+D:R=B*C*D;         #R=3L+3S+2A+2M
                                                   #A=5(L+A+M+S)
  3[5]:2@A=B*C+D:2@R=B*C+D;                        #R=2(L+A+M+S)
                                                   #A=2(L+A+M+S)
  4[5]:3@R=B/C:R=COS[B]:R=SIN[B];                  #R=5L+2COS+3D+5S
  5[3]:#A=CPX[B,C];                                #A=3(L+CPX+S)
  6[176]:2@A=CPX[B,C]:76@R=B*C                     #R=77L+76M+77S
        :48@A=B*C:48@A=B:I=B;                      #A=98L+2CPX+48M+96S
  7[24]:A=B*C+D+E;                                 #A=24(L+2A+M+S)
  8[1]:WHILE ITTER<K20 DO                          #R=20((5+11N)T+(53+58N)(L+S)+A
                                                      +(2N)M+D+#R[SOLVE]+48CABS
                                                      +(1+N)#R[CALC1]+(N)#R[CALC2]
                                                      +(2N)IMAG
                                                   #A=20((582+44N)(L+S)+(96+27N)A
                                                      +(6+31N)M+(N)D+48CPX+#A[SOLVE]
       BEGIN                                          +(1+N)#A[CALC1]+(N)#A[CALC2]
         1[2]:I=B+C:I=B;                            #R=2L+A+2S
         2[8]:6@A=B*C:CALC1:                        #R=#R[CALC1]+D+T+3L+3S
              IF MOD[ITTER,4]<>1 THEN               #A=#A[CALC1]+582(L+S)+96A+6M
               2(T):L=B ELSE                           +48CPX
               2(E)[577]:L=B:480@A=B:
               :48@A=B-C:48@A=B-CPX[0,C];
         3[1]:FOR J:=1..N DO                        #R=N(3T+54L+54S+2M+2IMAG
                                                      +#R[CALC1]+#R[CALC2])
                                                    #A=N(44L+44S+27A+31M+D
                                                      +#A[CALC1]+#A[CALC2])
              BEGIN
                1[5]:2@R=B*C:2@A=B*C                   #R=T+#R[CALC1]+2(L+M*S)
                    :IF J<N THEN CALC1                 #A=3(L+M+S)+#A[CALC1]
                    ELSE 1(E)[6]:A=B*C;
                2[2]:A=B*C;                            #A=2(L+M+S)
                3[24]:A=B*C;                           #A=24(L+M+S)
                4[2]:A=-(B+C+D+E+F+G+H+I)              #A=2(L+13A+M+S)
                    :A=B*(C+D+E+F+G+H);
                5[1]:A=-(B+C+D+E+F+G+H+I+J-K)/M        #A=L+A+D+S
                6[1]:IF LOC(J) THEN CALC2;             #R=T+#R[CALC2]
                                                       #A=#A[CALC2]
                7[64]:12@A=B:26@R=REAL[B]              #R=52(L+IMAG+S)
                     :26@R=IMAG[B]                     #A=12(L+S)
              END;
         4[2]:L=B:SOLVE                             #R=L+S+#R[SOLVE]
                                                    #A=#A[SOLVE]
         5[48]:R=CABS[B];                           #R=48(L+CABS+S)
         6[2]:FOR I:=1..2 AND J:=1..N DO            #R=4N(2T+L+S)
              IF ABSU(I,J)>ERR THEN R=B
              :FOR I:=1..2 AND J:=1..N DO
              IF ABSUDOT(I,J)>KE10 THEN L=B;
         7[1]:IF CONVERGE THEN I=B                  #R=T+L+S
       END {S8}
END {NEWRAP}
```

```
PROCEDURE CALC1                 #R=T+2L+2S+CABS
BEGIN {CALC1}                   #A=10L+20M+10A+10S
  1[10]:A=B+C*D*E;                          #A=10L+20M+10A+10S
  2[1]:R=CABS[B];                           #R=L+CABS+S
  3[1]:IF ABSR>RMAX THEN R=B                #R=T+L+S
END; {CALC1}
PROCEDURE CALC2                 #R=4T+3L+3S+6A+13#R[EXTER]
BEGIN {CALC2}                   #A=50L+50S+98A+24M+24D+13#A[EXTER]
  1[1]:EXTER;                               #R=#R[EXTER]
                                            #A=#A[EXTER]
  2[2]:A=B+C;                               #A=2L+2A+2S
  3[1]:IF FLAG THEN FOR L:=1..3 DO          #R=T+3(T+L+S+2A+4#R[EXTER])
       BEGIN                                #A=3(16L+16S+32A+8M+8D+4#A[EXTER])
         1[1]:I=B+C-D;                        #R=L+2A+S
         2[8]:A=B+C*(D+E);                    #A=8(L+2A+M+S)
         3[4]:EXTER;                          #R=4#R[EXTER]
                                              #A=4#A[EXTER]
         4[8]:A=B+(C-D)/E                     #A=8(L+2A+D+S)
       END
END; {CALC1}
```

```
PROCEDURE EXTER                    #R=(1/24)(49T+6L+6S+3A+3D+3CABS+2#R[JOURNL])
BEGIN {EXTER}                      #A=(1/24)(5L+5S+2A+8M+3CPX+2#A[JOURNL])
  1[2]:A=B;                                  #A=L+S
  2[1]:IF (J=5 OR J=19) THEN                 #R=T+(2/24)#R[JOURNL]
       BEGIN {2(T)}                          #A=(2/24)(L+A+M+S+#A[JOURNL])
         1[1]:JOURNL;
         2[1]:A=B-C*D
       END {2(T)} ELSE
       IF (J=3 OR J=12 OR J=21) THEN         #R=(22/24)T+(3/24)(T+2L+2S+A+D+CABS)
       BEGIN {2(E)(T)}                       #A=(3/24)(L+2M+CPX+S)
         1[1]:R=CABS[RR];                           #R=L+CABS+S
         2[1]:IF DEL>K002 THEN                      #R=T+L+A+D+S
              BEGIN {2(T)}                          #A=L+2M+CPX+S
                1[1]:R=B-C/D;
                2[1]:A=B*C*CPX[D,E]
              END {2(T)}
        END {2(E)(T)}
END; {EXTER}
```

```
PROCEDURE JOURNL                #R=3T+4X+5L+4S+A+M+CABS+SQRT+ATAN
BEGIN {JOURNL}                  #A=28X+13L+13S+9A+13M+5D+ABS+CONJ
  1:A=(B-C*D*E)/F;                    #A=X+L+A+2M+D+S
  2:R=CABS[V];                      #R=X+L+CABS+S
  3:IF ABSV<>0 THEN                  #R=3T+3X+4L+3S+A+M+SQRT+ATAN
    BEGIN {3(T)}9                    #A=27X+12L+12S+8A+11M+4D+ABS+CONJ
      1:A=ABS[B]*C/D;                    #A=X+L+M+D+ABS+S
      2:A=CONJ[B];                     #A=X+L+CONJ+S
      3:R=B*C;                           #R=X+L+M+S
      4:IF H<=KDM12 ... ELSE             #R=2T+2X+3L+2S+A+SQRT+ATAN
        BEGIN {4(E)}                     #A=25X+10L+10S+8A+10M+3D
          1:A=B*C*D;                         #A=X+L+2M+S
          2:R=SQRT(1-B);                   #R=X+L+A+SQRT+S
          3:A=(1-B)*C;                       #A=X+L+A+M+S
          4[4]:A=B*C;                        #A=4X+L+M+S
          5[5]:A=B*C;                        #A=5X+L+M+S
          6[4]:A=B*C;                        #A=4X+L+M+S
          7[5]:A=(B+C)/D;                    #A=5X+L+A+D+S
          8[2]:A=B-C*D/E;                    #A=2X+L+A+M+D+S
          9:A=B+C*D/E;                       #A=X+L+A+M+D+S
          10:IF NOT CAV THEN ... ELSE        #R=T+X+2L+S+ATAN
             BEGIN {10(E)}                   #A=2X+2L+4A+2M+2S
               1:R=ATAN[-B,REAL[C]];           #R=X+2L+ATAN+S
               2:A=B+C-D;                        #A=X+L+2A+S
               3:A=B*(C*D+E-F)                   #A=X+L+2A+2M+S
             END { 0(E)}
        END {4(E)}
   END {3(T)}
END;{JOURNL}
```

```
PROCEDURE SOLVE                 #R=96T+4909L+4909S+4098A+4098M+719D
BEGIN {SOLVE}                   #A=48L+48S+48A+48M+48CPX
  1:FOR J:=1..N DO                   #R=(N-1)(2T+182L+182S+152A+152M+30D)
    BEGIN                               +2T+70L+70S+41A+41M+29D
      IF J<N THEN                         #R=T+182L+182S+152A+152M+30D
      BEGIN {1(T)}
        1[9]:R=B/C;                            #R=9(L+D+S)
        2[63]:R=B-C*D;                         #R=63(L+A+M+S)
        3[8]:R=B/C;                            #R=8(L+D+S)
        4[48]:R=B-C*D;                         #R=48(L+A+M+S)
        5[7]:R=B/C;                            #R=7(L+D+S)
        6[17]:R=B-C*D;                         #R=17(L+A+M+S)
        7[6]:R=B/C;                            #R=6(L+D+S)
        8[24]:R=B-C*D                          #R=24(L+A+M+S)
      END {1(T)} ELSE                     #R=70L+70S+41A+41M+29D
      BEGIN {1(E)}
        S1[9]:R=B/C;
        S2[18]:R=B-C*D;
        S3[7]:R=B/C;
        S4[16]:R=B-C*D;
        S5[7]:R=B/C;
        S6[7]:R=B-C*D;
        S7[6]:R=B/C;
      END {1(E)}
    END; {1}
  2:FOR J:=N..1 DO              #R=(N-1)(2T+28L+28S+24A+24M)+2T+9L+9S+9A+9M
    BEGIN {2}                   #A=2N(L+S+A+M+CPX)
      IF J<N THEN                   #R=T+28L+28S+24A+24M
      BEGIN {2(T)}                  #A=2(L+A+M+CPX+S)
        1[16]:R=B*C;                     #R=16(L+M+S)
        2[4]:R=B-C-D-E-F;                #R=4(L+4A+S)
        3[3]:R=B-C*D;                    #R=3(L+A+M+S)
        4[2]:R=B-C*D;                    #R=2(L+A+M+S)
        5:R=B-C*D;                       #R=L+A+M+S
        6[2]:A=B-CPX[C*D]                #A=2(L+A+M+CPX+S)
      END {2(T)} ELSE               #R=9(L+A+M+S)
      BEGIN {2(E)}                  #A=2(L+A+M+CPX+S)
        S1[3]:R=B-C*D;
        S2[2]:R=B-C*D;
        S3[1]:R=B-C*D;
        S4[2]:A=B-CPX[C*D]
      END {2(E)}
  END {2}
END; {SOLVE}
```

APPENDIX D: THE CRITICAL PATH

```
PROGRAM SHAFT
BEGIN
  1:READ
  2:INITIALIZE;
  3:RESET;
  4:NEWRAP;
  5:IF CONVERGED THEN
    BEGIN
      1[2]:L=B;
      2:WHILE NOT STOP DO
              CALCULATE
    END
END;
PROCEDURE INITIALIZE
BEGIN
  1[206]:R=B*C;
  2[148]:R=B*C;
  3[96]:R=B/(C*(D-E));
  4[120]:R=B+C;
  5[48]:R=B+C;
  6[48]:R=B*C;
  7[24]:R=B-C;
  8[96]:R=B/C;
  9[120]:R=B
END; {INITIALIZE}
PROCEDURE RESET
BEGIN
  1[48]:A=B;
  2[170]:R=B/C;
  3[852]:R=-B*C,A=IMAG[R];
END; {RESET}
PROCEDURE CALCULATE
BEGIN
  1:IF CURRENT ... ELSE
    BEGIN {1E}
      WHILE T<=TMAX DO
      BEGIN
        1:IF CONVERGED AND Q<4 THEN
            1(E)[49]:A=B/C;
        2:CASE Q OF...FORESTEP4;
        3:NEWRAP
        4:IF NOT CONVERGED OR ERR>E04 ... ELSE
              IF ERR<E06 THEN  STEP§DOUBLE};
      END {1(E).1}
    END; {1(E)}
  2[3]:R=B;
  3:WRITE
END; {CALCULATE}
```

```
PROCEDURE FORESTEP4
BEGIN
  1[96]:A=B+C+D+E
END;
PROCEDURE NEWRAP          #R=14904X+1482L+1482S+3400T+21A+COS+20CABS
                             +480IMAG+480#R[CALC1]+480#R[CALC2]+20#R[SOLVE]
                          #A=13545X+1945L+1945S+7224A+1444M+480D+21CPX
                             +480#A[CALC1]+480#A[CALC2]+20#A[SOLVE]

BEGIN {NEWRAP}
  1[3]:R=B+C;                                  #R=3X+L+A+S
  2[5]:A=B*C+D;                                #A=5X+L+M+A+S
  3[3]:A=B*C+D;                                #A=3X+L+M+A+S
  4:R=COS[B];                                  #R=X+L+COS+S
  5[3]:A=CPX[B,C];                             #A=3X+L+CPX+S
  6[50]:A=B*C;                                 #A=50X+L+M+S
  7[24]:A=B*C+D+E;                             #A=24X+L+M+2A+S
  8:WHILE ITTER<K20 DO                         #R=20((2+7N)T+(49+29N)X+(2+3N)L+(2+3N)S
                                                     +A+(N)IMAG+CABS+#R[SOLVE]
    BEGIN                                            +(N)(#R[CALC1]+#R[CALC2]))
                                               #A=20((1+28N)X+(1+4N)L+(1+4N)S+(1+15N)A
                                                     +(3N)M+(N)D+#A[SOLVE]+CPX
                                                     +(N)(#A[CALC1]+#A[CALC2]))
      1:I=B+C;                                       #R=X+L+A+S
      2:IF MOD[ITTER,4]<>1 ... ELSE                  #R=T
          A=B-CPX[C];                                #A=X+L+CPX+A+S
      3:FOR J:=1..N DO                               #R=N(3T+27X+L+S
        BEGIN                                           +#R[CALC1]+#R[CALC2]+IMAG)
                                                     #A=N(28X+4L+4S+15A+3M+D
                                                        +#A[CALC1]+#A[CAL2])
          1:IF J<N THEN CALC1;                             #R=T+#R[CALC1]
                                                           #A=#A[CALC1]
          2[2]:A=B*C;                                      #A=2X+L+M+S
          3[24]:A=B*C;                                     #A=24X+L+M+S
          4:A=B*(C+D+E+F+G+H);                             #A=X+L+5A+M+S
          5:A=-(B+C+D+E+F+G+H+I+J-K)/M                     #A=X+L+10A+D+S
          6:IF LOC(J) THEN CALC2;                          #R=X+T+#R[CALC2]
                                                           #A=#A[CALC2]
          7[26]:R=IMAG[B]                                  #R=26X+L+IMAG+S
        END;
      4:SOLVE                                        #R=#R[SOLVE]
                                                     #A=#A[SOLVE]
      5[48]:R=CABS[B];                              #R=48X+L+CDABS+S
      6:FOR I:=1..2 AND J:=1..N DO                   #R=2N(2T+X+L+S)
        IF ABSU(I,J)>ERR THEN R=B;
      7:IF CONVERGE THEN I=B                         #R=T+X+L+S
    END {S8}
END;{NEWRAP}
PROCEDURE CALC1           #R=T+2X+2L+2S+CABS
BEGIN {CALC1}             #A=10X+L+S+A+2M
  1[10]:A=B+C*D*E;                #A=10X+L+2M+A+S
  2:R=CABS[B];                    #R=X+L+CABS+S
  3:IF ABSR>RMAX THEN R=B         #R=T+X+L+S
END; {CALC1}
```

```
PROCEDURE CALC2          #R=4T+3X+3L+3S+6A+4(#R[EXTER])
BEGIN {CALC2}            #A=50X+7L+7S+13A+3M+3D+4(#A[EXTER])
  1:EXTER;                        #R=#R[EXTER]
                                  #A=#A[EXTER]
  2[2]:A=B+C;                     #A=2X+L+A+S
  3:IF FLAG THEN FOR L:=1..3 DO   #R=T+3(T+X+L+S+2A+#R[EXTER])
    BEGIN                         #A=3(16X+2L+2S+4A+M+D+#A[EXTER])
      1:I=B+C-D;                        #R=X+L+2A+S
      2[8]:A=B+C*(D+E);                 #A=8X+L+2A+M+S
      3[4]:EXTER;                       #A=#A[EXTER]
                                        #R=#R[EXTER]
      4[8]:A=B+(C-D)/E                  #A=8X+L+2A+D+S
    END {3}
END; {CALC2}
```

```
PROCEDURE EXTER                #R=(1/24)(49T+2#R[JOURNL]+6X+6L+6S+3A+3D+3CABS)
BEGIN {EXTER}                  #A=(1/24)(53X+29L+29S+2A+8M+2#A[JOURNL]+3CPX)
  1[2]:A=B;                                  #A=2X+L+S
  2:IF (J=5 OR J=19) THEN                    #R=T+(2/24)#R[JOURNL]
    BEGIN {2(T)}                             #A=(2/24)(X+L+A+M+S+#A[JOURNL])
      1:JOURNL;
      2:A=B-C*D
    END {2(T)} ELSE
    IF (J=3 OR J=12 OR J=21) THEN  #R=(22/24)T+(3/24)(T+2X+2L+2S+A+D+CABS)
    BEGIN                                    #A=(3/24)(X+L+S+2M+CPX)
      1:R=CABS                                  #R=X+L+CABS+S
      2:IF DEL>K002 THEN                         #R=T+X+L+A+D+S
        BEGIN                                    #A=X+L+2M+CPX+S
          1:R=B-C/D;
          2:A=B*C*CPX
        END
    END
END; {EXTER}
PROCEDURE JOURNL             #R=3T+4X+5L+4S+A+M+CABS+SQRT+ATAN
BEGIN {JOURNL}               #A=28X+13L+13S+9A+13M+5D+ABS+CONJ
  1:A=(B-C*D*E)/F;                 #A=X+L+A+2M+D+S
  2:R=CABS[V];                    #R=X+L+CABS+S
  3:IF ABSV<>0 THEN                #R=3T+3X+4L+3S+A+M+SQRT+ATAN
    BEGIN {3(T)}                   #A=27X+12L+12S+8A+11M+4D+ABS+CONJ
      1:A=ABS[B]*C/D;                 #A=X+L+M+D+ABS+S
      2:A=CONJ[B];                   #A=X+L+CONJ+S
      3:R=B*C;                         #R=X+L+M+S
      4:IF H<=KDM12 ... ELSE           #R=2T+2X+3L+2S+A+SQRT+ATAN
        BEGIN {4(E)}                   #A=25X+10L+10S+8A+10M+3D
          1:A=B*C*D;                      #A=X+L+2M+S
          2:R=SQRT(1-B);                 #R=X+L+A+SQRT+S
          3:A=(1-B)*C;                    #A=X+L+A+M+S
          4[4]:A=B*C;                     #A=4X+L+M+S
          5[5]:A=B*C;                     #A=5X+L+M+S
          6[4]:A=B*C;                     #A=4X+L+M+S
          7[5]:A=(B+C)/D;                 #A=5X+L+A+D+S
          8[2]:A=B-C*D/E;                 #A=2X+L+A+M+D+S
          9:A=B+C*D/E;                    #A=X+L+A+M+D+S
          10:IF NOT CAV THEN ... ELSE     #R=T+X+2L+S+ATAN
             BEGIN {10(E)}                #A=2X+2L+4A+2M+2S
               1:R=ATAN[-B,REAL[C]];        #R=X+2L+ATAN+S
               2:A=B+C-D;                      #A=X+L+2A+S
               3:A=B*(C*D+E-F)                 #A=X+L+2A+2M+S
             END {10(E)}
        END {4(E)}
   END {3(T)}
END;{JOURNL}
```

```
PROCEDURE SOLVE                 #R=48X+4888X+312L+312S+264A+192M+96D
BEGIN {SOLVE}                   #A=48X+24L+24A+24M+24S+24CPX
  1:FOR J:=1..N DO                   #R=N(T+182X+8L+8S+4A+4M+4D)
    BEGIN
      IF J<N THEN                          #R=T+182X+8L+8S+4A+4M+4D
      BEGIN {1(T)}
        1[9]:R=B/C;                            #R=9X+L+D+S
        2[63]:R=B-C*D;                         #R=63X+L+A+M+S
        3[8]:R=B/C;                            #R=8X+L+D+S
        4[48]:R=B-C*D;                         #R=48X+L+A+M+S
        5[7]:R=B/C;                            #R=7X+L+D+S
        6[17]:R=B-C*D;                         #R=17X+L+A+M+S
        7[6]:R=B/C;                            #R=6X+L+D+S
        8[24]:R=B-C*D                          #R=24X+L+A+M+S
      END {1(T)} ELSE
      BEGIN {1(E)}                   NOTE: #R & #A FOR THIS ELSE CLAUSE ARE
        S1[9]:R=B/C;                       APPROXIMATELY ACCOUNTED FOR BY
        S2[18]:R=B-C*D;                    INCREASING THE THEN-CLAUSE-MULTIPLIER
        S3[7]:R=B/C;                       FROM N-1 TO N.
        S4[16]:R=B-C*D;
        S5[7]:R=B/C;
        S6[7]:R=B-C*D;
        S7[6]:R=B/C;
      END {1(E)}
    END; {1}
    FOR J:=1 TO N DO             #R-N(T+26X+5L+5S+7A+4M)
    BEGIN {2}                    #A=N(2X+L+A+M+S+CPX)
      IF J<N THEN                      #R=T+26X+5L+5S+7A+4M
      BEGIN {2(T)}                     #A=2X+L+A+M+CPX+S
        1[16]:R=B*C;                         #R=16X+L+M+S
        2[4]:R=B-C-D-E-F;                    #R=4X+L+4A+S
        3[3]:R=B-C*D;                        #R=3X+L+A+M+S
        4[2]:R=B-C*D;                        #R=2X+L+A+M+S
        5:R=B-C*D;                           #R=X+L+A+M+S
        6[2]:A=B-CPX[C*D]                    #A=2X+L+A+M+CPX+S
      END {2(T)} ELSE
      BEGIN {2(E)}                       NOTE: SEE NOTE ABOVE
        S1[3]:R=B-C*D;
        S2[2]:R=B-C*D;
        S3[1]:R=B-C*D;
        S4[2]:A=B-CPX[C*D]
      END {2(E)}
  END {2}
END; {SOLVE}
```

REFERENCES

1. Kascak, Albert F.: Direct Integration of Transient Rotor Dynamics. NASA TP-1597, 1980.

2. Arpasi, Dale K.; and Milner, Edward J.: Partitioning and Packing Mathematical Simulation Models for Calculation on Parallel Computers. NASA TM-87170, 1986.

TABLE I. - EXECUTION TIME ESTIMATES OF MC68020 MACHINE OPERATIONS

Machine operations		Execution time, μs	
Function	Imneumonic	Real result	Complex result
Load, store	L, S	0.32	0.64
Test and jump	T	2.64	----
Overhead	X	----	----
Add, sub, neg	A	4.0	8.0
Multiply	M	5.0	10.0
Divide	D	7.5	15.0
Square root	SQRT	8.5	----
Cosine	COS	38.5	----
Arctangent	ATAN	31.0	----
Abs. value	CABS	11.3	----
Abs. value	ABS	----	4.5
Convert	CPX, IMAG	1.3	1.3
Conjugate	CONJ	----	4.7

TABLE II. - OPERATIONAL SUMMARY FOR SERIAL CALCULATIONS OF THE NEWTON-RAPHSON ALGORITHM

(a) Operations required for real number results.

Procedure	T	L	S	A	M	D	COS	CABS	IMAG	SQRT	ATAN
NEWRAP main	5 380	28 991	28 991	28	1 040	23	2	960	960	0	0
500*CALC1	500	1 000	1 000	0	0	0	0	500	↓	↓	↓
480*CALC2	1 920	1 440	1 440	2 880	0	0	↓	0	↓	↓	↓
6240*EXTER	12 740	1 560	1 560	780	0	780	↓	780	↓	↓	↓
520*JOURNL	1 560	2 600	2 080	520	520	0	↓	520	↓	520	520
20*SOLVE	960	97 260	97 260	81 040	81 040	14 380	0	0	↓	0	0
NEWRAP	23 060	132 851	132 331	85 248	82 600	15 183	2	2760	960	520	520

(b) Operations required for complex number results

Procedure	L	S	A	M	D	CPX	CONJ	ABS
NEWRAP main	32 893	32 893	14 936	15 082	480	963	0	0
500*CALC1	5 000	5 000	5 000	10 000	0	0	↓	↓
480*CALC2	24 000	24 000	47 040	11 520	1520	0	↓	↓
6240*EXTER	1 300	130	520	2 080	0	780	↓	↓
520*JOURNL	6 760	6 760	4 630	6 760	2600	0	520	520
20*SOLVE	960	960	960	960	0	960	0	0
NEWRAP	70 913	69 743	73 086	46 402	4600	2703	520	520

TABLE III. - OPERATIONAL SUMMARY FOR THE CRITICAL PATH CALCULATION OF THE NEWTON-RAPHSON ALGORITHM

(a) Operations required for real number results.

Procedure	X	T	L	S	A	M	D	COS	CABS	IMAG	SQRT	ATAN
NEWRAP main	14 904	3 400	1 482	1 482	21	0	0	2	20	480	0	0
480*CALC1	960	480	960	960	0	0	0	0	480	0	↓	↓
480*CALC2	1 440	1 920	1 440	1 440	2880	↓	0	↓	0	↓	↓	↓
1920*EXTER	480	3 920	480	480	240	↓	240	↓	240	↓	↓	↓
160*JOURNL	640	480	800	640	160	160	0	↓	160	↓	160	160
20*SOLVE	97 760	960	6 240	6 240	5280	3840	1920	↓	0	0	0	0
NEWRAP	116 184	11 160	11 402	11 242	8581	4000	2160	2	900	480	160	160

(b) Operations required for complex number results

Procedure	X	L	S	A	M	D	CPX	CONJ	CONJ
NEWRAP main	13 545	1 945	1 945	7 224	1444	480	1	0	0
480*CALC1	4 800	480	480	480	960	0	0	↓	↓
480*CALC2	24 000	3 360	3 360	6 240	1440	1440	0	↓	↓
1920*EXTER	4 240	2 320	2 320	160	640	0	240	↓	↓
160*JOURNL	4 480	2 080	2 080	1 440	2080	800	0	160	160
20*SOLVE	960	480	480	480	480	0	480	0	0
NEWRAP	52 025	10 665	10 665	16 024	7044	2720	721	160	160

TABLE IV. - CALCULATION TIMES FOR PARALLEL AND SERIAL COMPUTATION OF THE NEWTON-RAPHSON ALGORITHM

(a) Serial computation

Procedure	Real # CALCS, msec	Complex # CALCS, msec	Total CALC, msec
NEWRAP main	50.41	320.86	371.27
500*CALC1	7.60	146.39	153.99
480*EXTER	17.51	695.30	712.81
6240*EXTER	52.41	27.63	80.04
20*SOLVE	36.71	157.70	194.41
520*JOURNL	901.99	19.75	921.74
		NEWRAP (total serial)	2434.26

(b) Parallel computation

Procedure	Real # CALCS, msec	Complex # CALCS, msec	Total CALC, msec
NEWRAP main	10.89 + 14904Tx	81.92 + 13545(2Tx)	92.81 + 41994 Tx
500*CALC1	7.30 + 906	14.50 + 4800	21.80 + 10506
480*EXTER	17.31 + 1440	90.22 + 24000	107.53 + 49440
6240*EXTER	16.12 + 480	10.96 + 4240	27.08 + 8960
20*SOLVE	11.29 + 640	48.45 + 4480	59.74 + 9600
520*JOURNL	61.24 + 97760	9.87 + 960	71.11 + 99680
		NEWRAP (total serial)	380.07 + 220180 Tx

where Tx and 2Tx are the assumed transfer times associated with real and complex number, respectively.

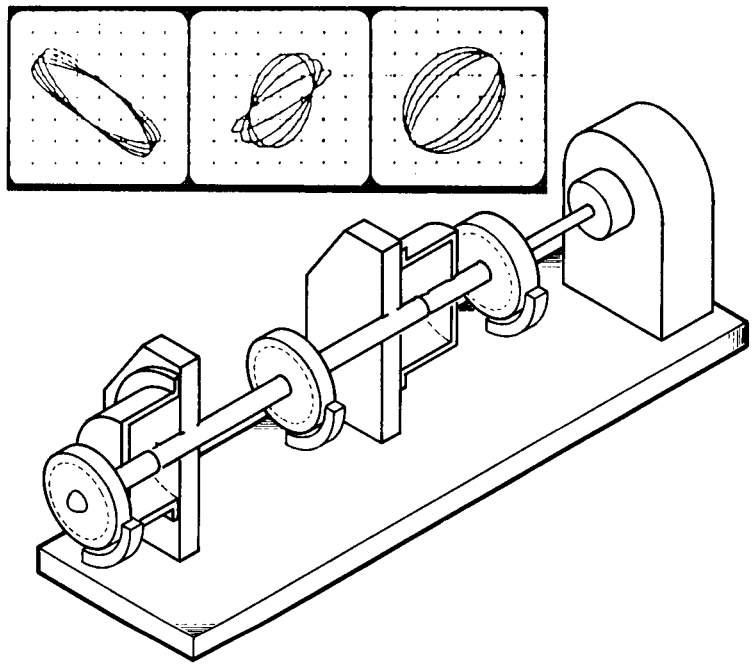

FIGURE 1. - THE ROTOR-BEARING SYSTEM.

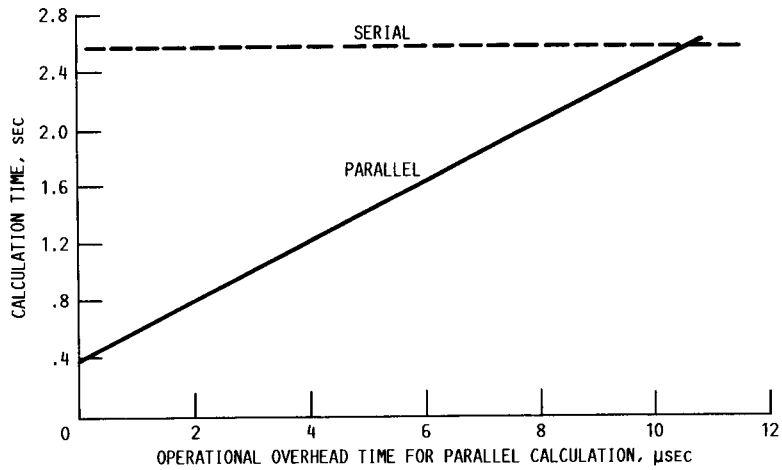

FIGURE 2. - EFFECT OF OPERATIONAL OVERHEAD ON PARALLEL CALCULATION TIME.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-101462		

4. Title and Subtitle	5. Report Date
Parallel Processing of a Rotating Shaft Simulation	February 1989
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Dale J. Arpasi	E-4290
	10. Work Unit No.
	505-62-21

9. Performing Organization Name and Address	11. Contract or Grant No.
National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191	
	13. Type of Report and Period Covered
	Technical Memorandum

12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Washington, D.C. 20546-0001	

15. Supplementary Notes

16. Abstract

A Fortran program describing the vibration modes of a rotor-bearing system is analyzed for parallelism, and a data-flow statement of the problem is developed. This statement identifies the inherent parallelism in this simulation using a pascal-like structured language. Potential vector operations are also identified. A critical path through the simulation is identified and used in conjunction with somewhat ficticious processor characteristics to determine the time to calculate the problem on a parallel processing system having those characteristics. A parallel processing overhead time is included as a parameter for proper evaluation of the gain over serial calculation. The serial calculation time is determined for the same ficticious system. An improvement of up to 640 percent is possible depending upon the value of the overhead time. Based on the analysis, certain conclusions are drawn pertaining to the development needs of parallel processing technology, and to the specification of parallel processing systems to meet specific computational needs.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Simulation Parallel processing	Unclassified – Unlimited Subject Category 66

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No of pages	22. Price*
Unclassified	Unclassified	68	A04

*For sale by the National Technical Information Service, Springfield, Virginia 22161