FINAL REPORT

AN OPTIMIZATION MODEL FOR THE U.S. AIR-TRAFFIC SYSTEM

John M. Mulvey

(NASA-CR-177277) AN OPTIMIZATION MODEL FOR THE US AIR-TRAFFIC SYSTEM Final Report (Princeton Univ., N. J.) 27 p HC A03/MF A01 CSCL 17G G3/04

N86-27271

Unclas 43135

School of Engineering and Applied Science
Princeton University
Princeton, New Jersey 08544

April, 1986

Grant No. NAG-1-520

Account No. 165-6210

National Aeronautics and Space Administration

An Optimization Model for the U. S. Air-Traffic System

John M. Mulvey

School of Engineering and Applied Science
Princeton University
Princeton, New Jersey 08544
April 1986

1. Executive Summary

This report summarizes the research carried out as part of the project "An Optimization Model for the U. S. Air-Traffic System (NASA# NAG-1-520). The primary research objective was to establish a systematic approach for monitoring U. S. air traffic in the context of system-wide planning and control. Towards this end, a network optimization model with nonlinear objectives was chosen as the central element in the planning/control system. The network representation was selected because: (1) it provides a comprehensive structure for depicting essential aspects of the air traffic system, (2) it can be solved efficiently for large scale problems, and (3) the design can be easily communicated to non-technical users through computer graphics.

Briefly, the network planning models consider the flow of traffic through a graph as the basic structure. Nodes depict locations and time periods for either individual planes or for aggregated groups of airplanes. Arcs define variables as actual airplanes flying through space or as delays across time periods. As such, a special case of the network can be used to model the so called flow control problem[2].

Due the large number of interacting variables and the difficulty in subdividing the problem into relatively independent subproblems, we designed an integrated model which will depict the entire high level (above 29000 feet) jet route system for the 48 contiguous states in the U. S. While the resulting model is large by today's standards, we felt that computer technology (hardware and software) is improving rapidly and a practical full-scale system should become feasible over the next few years.

As a first step in demonstrating the concept's feasibility, we gathered data from the Indianapolis control sector and built a nonlinear risk/cost model for this airspace. The nonlinear network program --NLPNETG-- was employed in solving the resulting test cases[1]. This optimization program uses the Truncated-Newton method (quadratic approximation)[3] for determining the search direction at each iteration in the nonlinear algorithm.

It was shown that aircraft could be re-routed in an "optimal" fashion whenever traffic congestion increased beyond an acceptable level, as measured by the nonlinear risk function. An efficient cost/risk frontier can be traced out by altering the relative weights between cost and risk. This curve can be employed within the context of strategic planning, e.g. adding airport resources, or as an operational decision tool for assisting air-traffic controllers.

Computational costs for this exercise were quite reasonable and the nonlinear optimization problems could be solved on an minicomputer -- VAX 11/750 --at Princeton University. In addition, we transported the optimization system to a CRAY supercomputer at Boeing Computer Services, Seattle, Washington. Taking advantage of the CRAY vector architecture required a substantial effort; see reference[6] for a description of the steps that were taken. This analysis demonstrated that the Trucated-Newton algorithm can be tuned for a vector computer and that substantial air-traffic control problems could be solved in real-time.

The technical details of the project are contained in three research reports. The report titles are listed below:

(1) "Nonlinear Network Programming on Vector Supercomputers," Report EES-85-13, Princeton University, submitted to *Operations Research*, 1986.

(2) "Real-Time Operational Planning for the U.S. Air-Traffic System", Report EES-86-5, Princeton University, submitted to *Applied Numerical Mathematics*, 1986.

(3) "Integrated Risk/Cost Planning Models for the U. S. Air Traffic System," Report EES-85-9, Princeton University, submitted to *Management Science*, 1985.

These reports are self-contained, describe the research work performed during the project, and are provided in the Appendix.

2. Future Research

We believe that future research involving integrated air-traffic modeling should be directed along several avenues. First, the size of the network test problems should be expanded by including multiple control sectors and a longer time horizon -- more periods. As the problems grow in size, every attempt must be made to keep the execution time within a reasonable level. The runtime issues are especially pertinent in the case of the operational planning model. Remember that the final goal is to develop a practical decision support system for air traffic planning.

Second, an interface should be designed so that non-technical personnel will be able to effectively use the planning systems. It will be essential to understand the type and the form of information which is needed by the air traffic controller, among others. An interactive procedure which employs graphics seems to be the most likely format. Perhaps, color and intensity could be used to identify the critical elements in the control domain. Also, the optimization procedure needs to be flexible enough to be able to handle alternative objective functions, priorities and possible intervention by the users. Criteria other than utilitarian can be achieved by adjusting the objective function in the network models.

Third, the risk objective function which measures issues such as congestion, proximity, and workload, must be refined and tested with empirical findings. There is much to be done in this arena. For example, the analysis must be coordinated with the micro-level studies carried out by Odoni[5] and others. The field of risk analysis is only beginning to develop general principles. As an approximation, one could use a congestion function which depends upon geometry, time of day, weather, and other factors to evaluate the degree of acceptability for an airspace as compared to delaying or re-routing planes anticipated to cross the region of interest. See the second and third papers listed in the Appendix for further details.

In summary, the National Airspace Plan[4] promises to greatly enhance the quality and the quantity of information available on the state of the U.S. air traffic system. Never the less, much effort will be needed to decide what information is critical to a safe and efficient air-traffic environment. In this regard, planning models will assist in important task of identifying critical informational flows. Both the strategic and the operational network models proposed in this project are designed for this task.

References

1. D. P. Ahlfeld, R. S. Dembo, J. M. Mulvey, and S. A. Zenios, "Nonlinear Programming on Generalized Networks," Report EES-85-7, submitted for publication to *Transactions on Mathematical Software*, Princeton University, June 1985.

2. M. Bielli, G. Calicchio, B. Nicolette, and S. Ricciavdelli, "The Air-traffic Flow Control Problem as an Application of Network Theory," *Computations and Operations Research*, vol. 9, no. 4, pp. 265-278, 1982.

3. R. S. Dembo, "A Primal Truncated Newton Algorithm for Large-Scale Nonlinear Network Optimization," School of Organization and Management Working Paper, Series B#72, Yale University, March 1983.

4. J. L. Helms, "Implementing the National Airspace System Plan," in *Safety Issues in Air Traffic Systems Planning and Design*, Princeton University Conference, Sept. 1983.

5. A. Odoni and S. Endoh, "A General Model for Predicting the Frequency of Air Conflicts," in *Safety Issues in Air Traffic Systems Planning and Design* , Princeton University , Sept. 1983.

6. S.A. Zenios and J.M. Mulvey, "Nonlinear Network Programming on Vector Supercomputers," Report EES-85-13, Princeton University, Feb., 1986.

APPENDIX

This appendix contains the computer program used for generating the Air-Traffic test cases in the project.

```
c*********************************************************************
c*********************************************************************
c**                                                               ***
c**                       PROGRAM FAANET                          ***
c**                                                               ***
c*********************************************************************
c*********************************************************************
c
c--- PURPOSE :
c             This program reads the data for the airports, flights and
c       fuel burn model and generates the network model for input to the
c       nonlinear network optimizer NLPNETG
c             It compiles with  F77 on the VAX 11/750 under UNIX 5.2
c
c--- SUBROUTINES :
c                 INPUT : Reads in the data for Airports/Flights/Fuel burn
c                 GENER     : Generates the network topology and identifies risk
c                 BODY      : Writes on the output file the generated network
c                 TAIL      : Writes on the output file the risk function
c
c                 LENGTH    : Calculates the distance traveled by a flight
c
c--- INPUT FILES :
c                 Logical unit 7   : Fuel burn model data
c                 Logical unit 8   : Airports location data
c                 Logical unit 9   : Flights strips data
c
c--- OUTPUT FILES:
c                 Logical unit 10  : Nonlinear network model
c                 Logical unit 15  : Warning/Error messages from the program
c                 Logical unit 16  : Scratch file for storage of interractions
c
c*********************************************************************
c
c--- DEFINITION OF VARIABLES
c
c... CHARACTER Arrays
c
c       airprt (nairpt)    : Four character airports code
c       flghts (nflight)   : Six character flight id code
c
c... REAL*4 Arrays
c
c       burn (nflight*2,9): Fuel burning rates at 9 legitimate cruise altitudes
c       cost (narc)        : Cost of generated network arcs
c
c... INTEGER Arays
c
c       iorig (nflight)    : Pointer to origin airport  for every flight
c       idest (nflight)    : Pointer to destination airport        "
c       iflght(nflight)    : Pointer to fuel burn record           "
c       alti  (nflight,2) : Entering/Exiting altitude             "
c       itime (nflight,2) : Entering time ( hour/min)             "
c       otime (nflight,2) : Exiting time  ( hour/min)             "
c
c       hemi  (nflight)    : Hemi code                             "
c       rate  (nflight)    : Burning rate indicator                "
c                                        - 1   Lbs/nmi
c                                        - 2   Lbs/hr
c
c       lati  (nairpt,3)   : Latitude of every airport
c       long  (nairpt,3)   : Longtitude of every airport
c       elev  (nairpt)     : Elevation of every airport
c
c       ii    (narc)       : From node of generated network arcs
c       jj    (narc)       : To node of generated network arcs
```

```
c
c       istart(9)           : Pointer to array ITAG for arcs contributing
c                                   to the risk function at different altitudes
c       itag  (narc )       : Thread of arcs contributing to risk
c
c... INTEGER Variables
c
c       nflight             : Number of flights
c       itper               : Number of time periods considered for flights delay
c       time                : Length of each delay time period
c       inter               : Number of time intervals considered for risk analysis
c
c***********************************************************************
c***********************************************************************
c-----------------------------------------------------------------------
c
        program faanet
c
c-----------------------------------------------------------------------
c
        include '../data/comdat'
c
        open (7)
        open (8)
        open (9)
        open (10)
        open (15)
        open (16)
        print *, ' [H [J'
c
c--- Read Input data
c
        write (15,900)
        call input
c
c--- Generate and write out the Network topology
c
        write (15,910)
        call gener
        call body
c
c--- Determine and write risk interractions over target sector
c
        write (15,920)
        call risk
        call tail
c
        print *, ' [H [J  #3 #4 D O N E !  [2B'
c
c--- FORMAT Statements ---------------------------------------
c
 900    format (/,1x,'*** READ INPUT DATA ...',/)
 910    format (/,1x,'*** GENERATE NETWORK TOPOLOGY ...',/)
 920  format (/,1x,'*** DETERMINE FLIGHT INTERRACTIONS ...',/)
c-----------------------------------------------------------
c
        close (7)
        close (8)
        close (9)
        close (15)
        close (16)
c
        stop
        end
c
c***********************************************************************
```

```
c
      subroutine input
c
c*********************************************************************
c
c--- PURPOSE :
c                       This subroutine reads the data from the three
c                       input files as described in the main program .
c
      include '../data/comdat'
      character*6 aflt
      character*4 aorig, adest
c
c--- Prompt the user for control parameters
c
      print *,'  [1;4;7m #3 #4 #6 FAA NETWORK GENERATOR  [0m [3B'
      print *,'Number of time periods considered for flight delay ?'
      read (5,*) itper
      print *,'Length of flight delay time interval ?'
      read (5,*) time
      print *,'Number of time periods considered for risk analysis ?'
      read (5,*) inter
      print *,'Value of printing flag ?'
      read (5,*) mprint
      print *,'Header card for generated network ( at most 72 char) ?'
      read (5,890) title
c
      print *,'  [3B [4C  [1;4;5;7m  #3 #4 #6 R U N N I N G  [0m'
c
c--- Read airports code and location
c
      i = 1
 10   read (8,900,end=20) airprt(i), lati(i,1),lati(i,2), lati(i,3),
     *                 long(i,1), long(i,2),long(i,3),elev (i)
      i = i + 1
      go to 10
c
c--- Read fuel burn data
c
 20   nairpt = i - 1
      i = 1
c
 30   read (7,910,end=40)flghts(i),hemi(i),rate(i),(burn(i,j),j=1,9)
      i = i + 1
      go to 30
c
c--- Read flights data and set pointers to airport and fuel-burn recors
c
 40   nfbrec = i - 1
      i = 1
c
 50   read (9,920,end=100) aflt, aorig, adest ,ialt1, ialt2 ,
     *                (itime(i,j),j=1,2), (otime(i,j), j=1,2)
c
      alti (i,1) = ialt1 * 100
      alti (i,2) = ialt2 * 100
c
      iflag1 = 1
      iflag2 = 1
c
c
      do 60 j = 1, nairpt
            if (aorig.eq.airprt(j)) then
                  iorig(i) = j
                  iflag1   = 0
            end if
```

```
                if (adest.eq.airprt(j)) then
                        idest(i) = j
                        iflag2   = 0
                end if
                if (iflag1.eq.0.and.iflag2.eq.0) go to 70
  60    continue
c
        if (iflag1.ne.0) write (15,930) aorig , aflt
        if (iflag2.ne.0) write (15,935) adest , aflt
c
  70    do 80 j = 1, nfbrec
                if (aflt.eq.flghts(j) ) then
                        iflght(i) = j
                        go to 90
                end if
  80    continue
c
        write (15,940) aflt
c
  90    i = i + 1
        go to 50
 100  nflight = i - 1
c
        if (mprint.le.3) go to 999
        write (15,950)
        write (15,960)
        write (15,970) (i,iorig(i),idest(i),
     *          (alti(i,j),itime(i,j),otime(i,j),j=1,2), i=1,nflight)
        write (15,980)
        write (15,990) (airprt(i), lati(i,1),lati(i,2),lati(i,3),
     *  long(i,1),long(i,2),long(i,3), elev(i), i=1,nairpt)
        write (15,992)
        write (15,994) (i,flghts(i),iflght(i),hemi(i),rate(i),
     *          burn(i,1),i=1,nfbrec)
c
c--- FORMAT Statements -------------------------------------------------
c
 890  format (a72)
 900    format (1x,a4,2(5x,3i5),5x,i5)
 910    format (1x,a6,14x,i1,1x,i1,9(1x,e11.0))
 920    format (1x,a6,6x,a4,1x,a4,5x,2(2x,i3),2(3x,i2),5x,2(3x,i2))
 930    format (1x,'No information available for origin airport ',a4,
     *        ' of flight ',a6)
 935    format (1x,'No information available for destination airport ',
     *          a4,' of flight ',a6)
 940    format (1x,'No information available for fuel burn rates of flight'
     *          1x,a6)
 950    format (1x,'--- OUTPUT from subroutine INPUT ---',/)
 960    format (/,1x,'No.',2x,'Orig',2x,'Dest',3x,'Altitude'
     *          ,2x,'In-tm',2x,'O-tm',/)
 970    format (1x,i3,1x,i5,1x,i5,1x,i4,2x,i4,1x,i2,2x,i2,1x,i2,2x,i2)
 980    format (/,1x,'Airprt',6x,'Latitude',8x,'Longitude',4x,'Elevation',/)
 990    format (2x,a4,2x,3i5,2x,3i5,3x,i5)
 992    format (1x,'No.',1x,' Flght',2x,'Iflght',1x,'Hemi',1x,'Rate',
     *          5x,'Fburn',/)
 994    format (1x,i3,1x,a6,2x,i5,3x,i1,4x,i1,2x,e15.6)
c-----------------------------------------------------------------------
c
 999    return
        end
c
c**********************************************************************
c
        block data
c
c**********************************************************************
```

```
c
      include '../data/comdat'
c
      data ih0 /29000,33000,37000,41000,45000/
      data ih1 /31000,35000,39000,43000/
      data nfaa/0/
c
      end
```

```
c
c*************************************************************************
c
        subroutine gener
c
c*************************************************************************
c
c--- PURPOSE :
c                       This subroutine generates the network topology for
c               the Flight/Airport information given for one control sector.
c               It generates alternative cruise altitudes for every flight
c               and determines associated flight costs as well as delay costs .
c                       For every generated route (time/altitude) its contribution
c               to the risk function is determined .
c
c--- INPUT   :
c                       The arrays read from the database files by routine INPUT
c                       ih0(nflight) , legitimate cruise altitudes for hemi-code = 0
c                       ih1(nflight) , legitimate cruise altitudes for hemi-code = 1
c
c--- OUTPUT  :
c                       ii (narc)          , from node of generated arc
c                       jj (narc)          , to node of generated arc
c                       cost   (narc)      , cost of generated arc
c                       ipntfl (narc)      , flight number of this arc
c                       iper   (narc)      , time period of this arc
c                       istart (inter*9)   , pointer to ITAG array
c                       itag (narc)        , thread of lfights contributing to risk
c                       narc               , total number of generated arcs
c
c
        include '../data/comdat'
c
        narc = 0
        i    = 0
        j    = 0
c
c=============================================================
c==   Consider origin nodes for all time periods.         ==
c=============================================================
c
        do 1000 i = 1, itper
c
                do 500 j = 1, nflight
c
                        iinod = (i-1)*nflight + j
                        jj1   = i*nflight + j
                        jj2   = (itper+i-1)*nflight + j
c
                        itin  = itime(j,1)*60 + itime(j,2) + (i-1)*time
                        itout = otime(j,1)*60 + otime(j,2) + (i-1)*time
c
c--- Takeoff delay arcs ...
c
                        ipnt  = iflght (j) + 1
                        if (rate(ipnt).ne.2) write (15,900) flghts(ipnt)
                        costd = time * burn(ipnt,1) / 60.0
c
c--- En route arcs ...
c
                        ipnt  = iflght (j)
                        ihcode= hemi  (ipnt)
                        ialt0 = 0
                        ialt1 = alti (j,1)
                        ialt2 = 0
c
```

```
c..... determine alternative cruise altitudes for this flight
c
                         if (ihcode.eq.1) go to 50
c
c..... alternatve routes for Hemi code 0 flights
c
                         if (alti(j,1) .eq. alti(j,2)) go to 30
                         ialt1 = ( alti(j,1) + alti(j,2)) / 2
c
                         do 20 k = 1, 4
                           if (ialt1.lt.ih0(k+1).and.ialt1.gt.ih0(k)) then
                                ialt1 = ih0(k+1)
                                go to 30
                           end if
  20                     continue
  30                     continue
c
                         do 40 k = 1, 5
                           if (ialt1.eq.ih0(k)) go to 45
  40                     continue
c
  45                     fburn1 = burn (ipnt,2*k-1)
                         if (fburn1.eq.0.0) write (15, 910) flghts(ipnt)
c
                         if (k.gt.1) then
                                ialt0 = ih0(k-1)
                                fburn0= burn (ipnt,2*k-3)
                                if (fburn0.eq.0.0) fburn0 = fburn1
                         end if
c
                         if (k.lt.5) then
                                ialt2 = ih0(k+1)
                                fburn2= burn (ipnt,2*k+1)
                                if (fburn2.eq.0.0) fburn2 = fburn1
                         end if
c
                         go to 100
c
c
c..... alternatve routes for Hemi code 1 flights
c
  50                     continue
                         if (alti(j,1).eq.alti(j,2)) go to 80
                         ialt1 = ( alti(j,1) + alti(j,2)) / 2
c
                         do 70 k = 1, 3
                           if (ialt1.lt.ih1(k+1).and.ialt1.gt.ih1(k)) then
                                ialt1 = ih1 (k+1)
                                go to 80
                           endif
  70                     continue
  80                     continue
c
                         do 90 k = 1, 4
                                if (ialt1.eq.ih1(k)) go to 95
  90                     continue
c
  95                     fburn1 = burn(ipnt,2*k)
                         if (fburn1.eq.0.0) write (15, 910) flghts(ipnt)
c
                         if (k.gt.1) then
                                ialt0 = ih1(k-1)
                                fburn0= burn (ipnt,2*k-2)
                                if (fburn0.eq.0.0) fburn0 = fburn1
                         end if
c
```

```
                        if (k.lt.4) then
                                ialt2 = ih1(k+1)
                                fburn2= burn (ipnt,2*k+2)
                                if (fburn2.eq.0.0) fburn2 = fburn1
                        end if
c
c..... determine the cost of alternative altitudes
c
 100                    continue
                        call length (j ,dist)
c
                        cost0 = fburn0 * dist
                        cost1 = fburn1 * dist
                        cost2 = fburn2 * dist
c
c==============================================================================
c==  Store generated arcs. (For the last time period no delays possible)      ==
c==============================================================================
c
c---> Save Delay arcs
c
                        if (i.ne.itper) then
                                narc = narc + 1
                                ii (narc) = iinod
                                jj (narc) = jj1
                                cost(narc)= costd
                                ipntfl(narc) = j
                                iper  (narc) = i
                        end if
c
c---> Save flight arcs on different altitudes
c
c--- PRIMARY-1 -----------------------------------
                        if (ialt0.gt.0) then
                                narc = narc + 1
                                ii (narc) = iinod
                                jj (narc) = jj2
                                cost(narc)= cost0
                                ipntfl(narc) = j
                                iper  (narc) = i
                        end if
c
c--- PRIMARY    ---------------------------------
                        narc = narc + 1
                        ii (narc) = iinod
                        jj (narc) = jj2
                        cost(narc)= cost1
                        ipntfl(narc) = j
                        iper  (narc) = i
c
c--- PRIMARY+1  ---------------------------------
                        if (ialt2.gt.0) then
                                narc = narc + 1
                                ii (narc) = iinod
                                jj (narc) = jj2
                                cost(narc)= cost2
                                ipntfl(narc) = j
                                iper  (narc) = i
                        end if
c
 500            continue
1000    continue
c
c=====================================================================
c==   Consider destination nodes for all time periods               ==
c=====================================================================
```

```
c
      itor   = itper * nflight
      itort2 = itor * 2.0
c
      do 2000 i = 1, itper
c
              do 1500 j = 1, nflight
c
                      iinod = itor + (i-1)*nflight + j
                      jj1   = itor +    i *nflight + j
                      jj2   = itort2               + j
c
c--- Landing delay arcs ...
c
                      ipnt = iflght (j) + 1
                      if ( rate(ipnt).ne.2) write (15,900) flghts(ipnt)
                      costd= time * burn(ipnt,1) / 60.0
c
c--- Store generated arcs. (For the last time period no delays possible)
c
                      if (i.ne.itper) then
                              narc = narc + 1
                              ii(narc)   = iinod
                              jj(narc)   = jj1
                              cost(narc)= costd
                              ipntfl(narc) = j
                              iper   (narc) = i
                      end if
c
                      narc = narc + 1
                      ii(narc)   = iinod
                      jj(narc)   = jj2
                      cost(narc)= 0.0
                      ipntfl(narc) = j
                      iper   (narc) = i
c
1500          continue
2000  continue
c
c--- FORMAT Statements -------------------------------------------------
c
 900  format (1x,'Check fuel burn data for flight ',a6)
 910  format (1x,'Fuel burn data for flight ',a6,' is not available')
c-----------------------------------------------------------------------
c
      return
      end
c
c***********************************************************************
c
      subroutine length (iflt, dist)
c
c***********************************************************************
c
c--- PURPOSE :
c             Returns the distance between two airports - origin and
c      destination - of a flight.
c             NOTE : The distance is measured ignoring difference in
c      the elevation of the origin-destination airports and the cruise
c      altitude of the flight. The earth curvature is taken into account.
c
c--- INPUT   :
c      iflt              : Flight number indicator
c      airprt (nairpt)   : List of airport codes
c      lati (nairpt,3)   : Latitude of all airports
c      long (nairpt,3)   : Logitude of all airports
```

```
c       iorig  (nflght)    : Pointer to origin airport for flight IFLT
c       idest  (nflght)    : Pointer to destination airport for flight IFLT
c
c       degrad             : PI/180 - converts degrees into radians
c         radmil             : 24844mls /2*PI*1.151 -converts rad into naut. miles
c
        include '../data/comdat'
        real lat1, lat2
c
c--- Parameter initialization
c
        parameter ( degrad = 0.017453 , radmil = 3439.4 )
        ipo  = iorig (iflt)
        ipd  = idest (iflt)
c
c-----------------------------------------------------------------------
c       Calculate latitudes of Origin/Destination airports in radians
c       Compute the linear distance between the airports as :
c         (GREAT CIRCLE DISTANCE) * (EARTH CIRCUITY FACTOR)
c-----------------------------------------------------------------------
c
        x1 = lati(ipo,1) + lati(ipo,2) / 60.0 + lati(ipo,3) / 3600.0
        y1 = long(ipo,1) + long(ipo,2) / 60.0 + long(ipo,3) / 3600.0
        x2 = lati(ipd,1) + lati(ipd,2) / 60.0 + lati(ipd,3) / 3600.0
        y2 = long(ipd,1) + long(ipd,2) / 60.0 + long(ipd,3) / 3600.0
c
        lat1 = degrad * y1
        lat2 = degrad * y2
c
        dist = acos(sin(lat1)*sin(lat2) + cos(lat1)*cos(lat2)*
     1          cos(degrad*(x1-x2)))*radmil
c
        if (mprint.ge.3) write (15,900) iflt, dist
c
c--- FORMAT Statements ---------------------------------------------------
c
 900    format (1x,'Length of flight ',i3,' is ',e25.12)
c-----------------------------------------------------------------------
c
        return
        end
```

```
c***********************************************************************
c
        subroutine body
c
c***********************************************************************
c
c--- PURPOSE :
c                This subroutine writes on the output file the body
c       of the generated network.
c
c--- INPUT    :
c                The arrays generated by routine GENER
c
c-----------------------------------------------------------------------
c
        include '../data/comdat'
c
c--- Initialization
c
        character*1 a, s, d
        character*2 bd
        parameter (i0=0, i1=1, i2=2, r0=0, r1=1)
        data a/1ha/, s/1hs/, d/1hd/, bd/2hbd/
c
c--- Check the size of generated network
c
        nodes = 2*nflight*itper + nflight
        na    = 3*nflight*itper + 2*nflight*(itper-1) + nflight*itper
        nn    = jj (narc)
c
        if (na.ne.narc)  write (15,900) na, narc
        if (nn.ne.nodes) write (15,910) nodes, nn
c
c--- Write header card
c
        write (10,920) title
        write (10,930) nn, narc, i2, i1
c
c--- Write arc data
c
        write (10,940) a
        do 100 i = 1, narc
                write (10,950) bd, i0, ii(i), jj(i),r0,r1,r0,cost(i),r0,r0
 100    continue
c
c--- Write supply data
c
        write (10,940) s
        do 200 i = 1, nflight
                write (10,970) i , r1
 200    continue
c
c--- Write demand data
c
        write (10,940) d
        itemp = 2*nflight*itper
        do 300 i = 1, nflight
                nod = itemp + i
                write (10,970) nod, r1
 300    continue
c
c--- FORMAT Statements ---------------------------------------------------
c
 900    format (1x,'WARNING : Model should have ',i5,' arcs. Only ',i5,
     *           ' were generated.',/,
     *11x,'Data for some flights may be missing. Check the databases',/)
```

```
 910    format (1x,'ERROR   : Model should have ',i3,' nodes. Only ',i3,
     *       ' were generated.',/,
     *9x,'Possible program error.',/)
 920    format (1x,a72)
 930    format (4i5)
 940    format (a1,t72,'  ')
 950    format (3x,a2,3x,i2,2i5,3f10.1,3f10.1)
 970    format (10x,i5,5x,f10.0,t72,'  ')
c
        return
        end
c
c**************************************************************************
c
        subroutine tail
c
c**************************************************************************
c
c--- PURPOSE :
c                This subroutine writes on the output file the components
c       of the risk function (ie. nonlinear and non-separable function).
c                For the special case when only a single flight is crossing
c       the sector the risk function is separable.
c
c--- INPUT   :  The arrays generated by routine RISK
c
c
        include '../data/comdat'
c
        character*3 f , e
        parameter (zero=0.0, two=2.0)
        data f/3hfaa/, e/3hend/
        imax   = - 1
        imin   = 99999
        rmxcst= 0.0
c
c---> Determine the max cost to use as scaling factor.
c
        do 5 i = 1, narc
                costi = cost (i)
                if (rmxcst.lt.costi ) rmxcst = costi
  5     continue
        rmxcst = rmxcst / 10.0
c
c=============================================================================
c===                 Determine the planning interval                       ===
c=============================================================================
c
        do 10 i = 1, nflight
                itmax = otime(i,1)*60 + otime(i,2)
                itmin = itime(i,1)*60 + itime(i,2)
                if (itmax.gt.imax) imax = itmax
                if (itmin.lt.imin) imin = itmin
  10    continue
c
c---> Length of interval used for risk evaluation
c
        rintl = real (imax-imin) / real (inter)
        write (15,900) imin, imax, rintl
c
c=============================================================================
c===              Consider all flights at the same cruise altitude            ===
c=============================================================================
c
        write (10,910) f
        do 1000 i = 1, 9
```

```
c
                 ipi = istart(i)
                 if (ipi.eq.0.or.itag(ipi).eq.0) go to 1000
c
 100             if (ipi.eq.0) go to 1000
                         ipj = ipi
 200                     ipj = itag (ipj)
                         if (ipj.eq.0) go to 500
c
                         iarci = ipntfl (ipi)
                         iarcj = ipntfl (ipj)
c
                         if (iarci.eq.iarcj) go to  200
c
                         iaddi = time   * (iper(ipi) - 1)
                         iaddj = time   * (iper(ipj) - 1)
                         isti   = itime(iarci,1)*60 + itime(iarci,2) + iaddi
                         iendi = otime(iarci,1)*60 + otime(iarci,2) + iaddi
                         istj   = itime(iarcj,1)*60 + itime(iarcj,2) + iaddj
                         iendj = otime(iarcj,1)*60 + otime(iarcj,2) + iaddj
c
                         ist    = max0 (isti ,istj )
                         iend   = min0 (iendi,iendj)
                         if (ist.ge.iend) go to 200
c
c---> ALPHA is proportional to the overlapping time in sector
c
                         rtemp = iend - ist
                         alpha = rmxcst * (rtemp / rintl)
                         write (16,920) ipi, ipj, alpha, two, zero
                         go to 200
c
 500             ipi = itag (ipi)
                 go to 100
c
1000     continue
c
         write (16,920) narc, narc, zero, zero, zero
         close (10)
         call system ('sort +10 -15 < fort.16 >> fort.10')
c        write (10,910) e
         if(mprint.ge.2) write(15,930)
      *   (istart(i),i=1,9) , (itag(i),i=1,narc)
c
c--- FORMATS -----------------------------------------------------------
 900    format (1x,'Planning interval: From ',i3,'min to ',i4,
      *    1x,'min. Length of risk interval:',f6.2)
 910     format (a3,t25,'1.0',t72,'    ')
 920     format (10x,2i5,3f10.4)
 930     format (1x,'ISTART(i)',9(1x,i5),/,1x,'ITAG(i)',/,100(10(1x,i5),/))
c
         return
         end
```

```
c**********************************************************************
c
      subroutine risk (num, ist)
c
c**********************************************************************
c
c--- PURPOSE :
c                This subrutine generates the arrays for the nonlinear
c       part of the network model. For a given flight/time it identifies the
c       altitude and time interval when a conflict may arise and creates the
c       thread that links all interracting flights
c
c--- INPUTS    :
c                num   , number of risk periods while flight is in target sector
c                ist   , pointer to the flight altitude while in target sector
c
c--- OUTPUTS   :
c                istart, pointer to the starting location of the thread of
c                          interracting flights
c                itag  , thread of interracting flights
c
c
      include '../data/comdat'
c
c==========================================================================
c=== Detemine the flights that cross the target sector at the same altitude ===
c==========================================================================
c
c---> Initialize storage arrays
c
                do 12 i = 1, 9
                        istart (i) = 0
 12             continue
                do 15 i = 1, narc
                        itag (i) = 0
 15             continue
c
                do 500 i = 1, itper
                do 200 j = 1, nflight
c
                        ipnt  = iflght (j)
                        ihcode= hemi (ipnt)
                        ialt0 = 0
                        ialt1 = alti (j,1)
                        ialt2 = 0
c
                        iinod = (i-1)*nflight + j
                        jj2   = (itper+i-1)*nflight + j
c
c---> Find all arcs for this flight
c
                        icount = 0
  18                    icount = icount + 1
                        if (ii(icount).eq.iinod.and.jj(icount).eq.jj2) goto 19
                        if (icount.lt.narc) go to 18
                        write (15,910) flghts(j), i
                        go to 200
c
  19                    continue
c
c---> Determine the flight altitude
c
                        if (ihcode.eq.1) go to 50
c
c... Altitude for Hemi code 0 flights
c
```

```
                        if (alti(j,1).eq.alti(j,2)) go to 30
                        ialt1 = (alti(j,1) + alti(j,2)) / 2
c
                        do 20 k = 1, 4
                          if (ialt1.lt.ih0(k+1).and.ialt1.gt.ih0(k)) then
                                ialt1 = ih0(k+1)
                                go to 30
                          end if
   20                   continue
   30                   continue
c
                        do 40 k = 1, 5
                          if (ialt1.eq.ih0(k)) go to 45
   40                   continue
   45                   continue
c
c---> Save all the arcs corresponding to this flight at all altitudes
c
c--- PRIMARY - 1 --------------------------------------------------
        if (k.ne.1.and.ii(icount).eq.iinod.and.jj(icount).eq.jj2) then
                        km1 = k - 1
                        call save (km1, icount)
                        icount = icount + 1
        end if
c--- PRIMARY     --------------------------------------------------
        if (ii(icount).eq.iinod.and.jj(icount).eq.jj2) then
                        call save (k,icount)
                        icount = icount + 1
        end if
c--- PRIMARY + 1 --------------------------------------------------
        if (k.ne.5.and.ii(icount).eq.iinod.and.jj(icount).eq.jj2) then
                        kp1 = k + 1
                        call save (kp1, icount)
        end if
c
                        go to 200
c
c... Altitude for Hemi code 1 flights
c
   50                   continue
                        if (alti(j,1).eq.alti(j,2)) go to 80
                        ialt1 = (alti(j,1) + alti(j,2)) / 2
c
                        do 70 k = 1, 3
                          if (ialt1.lt.ih1(k+1).and.ialt1.gt.ih1(k)) then
                                ialt1 = ih1(k+1)
                                go to 80
                          end if
   70                   continue
   80                   continue
c
                        do 90 k = 1, 4
                          if (ialt1.eq.ih1(k)) go to 95
   90                   continue
   95                   continue
c
c---> Save all the arcs corresponding to this flight at all altitudes
c
c--- PRIMARY - 1 --------------------------------------------------
        if (k.ne.1.and.ii(icount).eq.iinod.and.jj(icount).eq.jj2) then
                        km1 = (k - 1) + 5
                        call save (km1, icount)
                        icount = icount + 1
        end if
c--- PRIMARY     --------------------------------------------------
        if (ii(icount).eq.iinod.and.jj(icount).eq.jj2) then
```

```
                         kp5 = k + 5
                         call save (kp5,icount)
                         icount = icount + 1
         end if
c--- PRIMARY + 1 --------------------------------------------------------
         if (k.ne.4.and.ii(icount).eq.iinod.and.jj(icount).eq.jj2) then
                         kp1 = (k + 1) + 5
                         call save (kp1, icount)
         end if
c
 200             continue
 500             continue
c
c--- FORMATS ------------------------------------------------------------
 905     format (1x,'Flight ',a6,' at time period',i3,
     *   ' is in target sector. Time in:',i3,' Time out:',i3)
 910     format (1x,'*** WARNING *** No arcs were generated for flight ',
     *     a6,' at time period ',i5)
c
         return
       end
c
c***********************************************************************
c
       subroutine save (ipos, iarcn)
c
c***********************************************************************
c
c--- PURPOSE :
c                        This subroutine stores the specified flight
c        in the array of all flights that are crossing the target sector
c        simultaneoulsy and at the same level
c
c--- INPUTS   :
c                        iarcn  , arc number for flight (time/alittude) considered
c                        ipos   , pointer to array ISTART indicating altitude
c--- OUTPUTS  :
c                        istart , pointer to thread storing all flights at this level
c                        itag   , thread of all flights crossing target sector at the
c                                    same interval
c
       include '../data/comdat'
c
         if (mprint.ge.3) write (15,900) iarcn, ipos
c
c--- Is this the first flight at this altitude ?
c
       ip = istart (ipos)
       if (ip.eq.0) then
             istart (ipos) = iarcn
             go to 9999
       end if
         ipm1 = ip
c
c--- If more flights at this altitude exist find an empty spot to store this
c
 100   ip = itag (ipm1)
       if (ip.eq.0) then
             itag(ipm1) = iarcn
             go to 9999
       end if
         ipm1 = ip
       go to 100
c
9999   if (mprint.ge.3) then
                 write (15,910) (istart(i),i=1,9)
```

```
                 write (15,920) (itag(i),  i=1,narc)
         end if
c
c--- FORMATS -----------------------------------------------------------------
 900     format (1x,'Save arc No. ',i5,' in altitude position ',i3)
 910     format (1x,'ISTART(i) :',9(1x,i5))
 920   format (1x,'ITAG(i)    :',/,100(10(1x,i5),/))
c
         return
       end
```

```
      character*28  char(1)
      character*48  char2(1), char3(1)
10    read (7,100,end=500) char(1)
      read (8,110) char2(1)
      read (9,110) char3(1)
      write (10,200) char(1)(2:8), char(1)(9:12), char(1)(13:14)
     *          ,char(1)(15:15),char(1)(16:16),char(1)(17:29)
     * ,char2(1)(1:13),char2(1)(13:25),char2(1)(25:37),char2(1)(37:49)
     * ,char3(1)(1:13),char3(1)(13:25),char3(1)(25:37),char3(1)(37:49)
 100  format (a28)
 110  format (a48)
 200  format (1x,a6,3x,2x,a3,3x,a2,2(1x,a1),9(a12))
      go to 10
c
500   print *, 'Finished'
      stop
      end
```

```
      character*21  char(1)
10    read (9,100,end=500) char(1)
      write (10,200) char(1)(1:4), char(1)(5:6), char(1)(7:8)
     *          ,char(1)(9:10),char(1)(11:13),char(1)(14:15)
     *          ,char(1)(16:17),char(1)(18:21)
100   format (a21)
200   format (1x,a4,5x,3(3x,a2),5x,2x,a3,2(3x,a2),5x,1x,a4)
      go to 10
c
500   print *, 'Finished'
      stop
      end
```

```
      character*29  char(1)
10    read (9,100,end=500) char(1)
      write (10,200) char(1)(1:6), char(1)(8:11), char(1)(12:15)
     *          ,char(1)(16:18),char(1)(19:21),char(1)(22:23)
     *          ,char(1)(24:25),char(1)(26:27),char(1)(28:29)
100   format (a29)
200   format (1x,a6,5x,2(1x,a4),5x,2(2x,a3),2(3x,a2),5x,2(3x,a2))
      go to 10
c
500   print *, 'Finished'
      stop
      end
```