Chapter 19

An Implementation of Multiple Class, Exact MVA

19.1. Introduction

In this appendix we provide a Fortran implementation of the exact mean value analysis algorithm for separable queueing network models consisting entirely of queueing centers and containing three classes of batch type.

The algorithm on which this program is based is described in Chapter 7. The interested reader will find it educational to extend the program in three ways: to allow more than three classes, to allow delay centers, and to allow a choice of batch, terminal, or transaction types independently for each class. The extension to load dependent service centers is discussed in Chapter 20.

As with the program in Chapter 18, our intention is that this program be used for educational experimentation with simple models. Its value as a capacity planning tool in no way approaches that of commercial queueing network modelling software. For a better idea of the interactions possible with that type of software, consult Chapter 16.

19.2. The Program

The program appears on the next four pages. Two statement labels (2001 and 2003) are included for reference in Chapter 20 and are not used in the program.

Note that some Fortran implementations impose restrictions on formatted I/O. It is best to include an explicit decimal point in real-valued input (but not integer-valued input) when using the program.

```
            program multpl
c
c A maximum of 3 classes and 25 centers are allowed.
c Classes 2 and 3 are limited to a maximum of 10 customers each.
c Class 1 has no limit on its population.
c
            integer Ncents,center,class
            integer n1,n2,n3
c
c Ncusts(c) is the population of class c.
c N is a temporary population vector required by MVA.
c
            integer Ncusts(3),N(3)
c
c demand(c,k) is the service demand of class c at center k.
c
            real demand(3,25)
c
c qlen(class 2 pop.,class 3 pop.,k) is a 3-dimensional array
c containing the queue length at each center k for each possible
c combination of class 2 and class 3 network populations. The
c population of class 1, the outermost class in the iteration,
c need not appear as an index. Population indices run from 1 to
c 11 to represent populations from 0 to 10 because some Fortran
c implementations restrict the base of array dimensions to be 1.
c
            real qlen(11,11,25)
c
c rtime(c,k) is the residence time of class c at center k.
c
            real rtime(3,25)
c
c tput(c) and sysr(c) are the throughput and response time of class c.
c
            real tput(3),sysr(3)
c
            write (6,5)
5           format (30h Input number of customers for)
            do 10 class = 1,3
                write (6,15) class
15              format (9h   Class ,i1,1h:)
                read (5,20) Ncusts(class)
20              format (i4)
10              continue
```

```
          write (6,25)
25        format (25h Input number of centers:)
          read (5,20) Ncents
          write (6,30)
30        format (25h Input service demand for)
          do 35 center=1,Ncents
              write (6,40) center
40            format (10h   Center ,i2,1h:)
              do 45 class=1,3
                  if (Ncusts(class) .eq. 0) goto 45
                  write (6,50) class
50                format (11h     Class ,i1,1h:)
                  read (5,55) demand(class,center)
55                format (f8.4)
45                continue
35            continue
c
c Now that the network is described, we perform the evaluation.
c The algorithm iterates through all possible population vectors.
c
          do 60 n1=0,Ncusts(1)
              do 65 n2=0,Ncusts(2)
                  do 70 n3=0,Ncusts(3)
                      if (n1+n2+n3 .eq. 0) goto 70
                      N(1) = n1
                      N(2) = n2
                      N(3) = n3
c
c First, compute the residence time at each center.
c
                      do 80 class = 1,3
                          sysr(class) = 0.0
                          if (N(class) .eq. 0) goto 80
c
                          N(class) = N(class) - 1
                          do 85 center=1,Ncents
2001                          rtime(class,center) =
     x                            demand(class,center) *
     x                            (1.0+qlen(N(2)+1,N(3)+1,center))
                              sysr(class) = sysr(class) +
     x                            rtime(class,center)
85                            continue
                          N(class) = N(class) + 1
```

```
c
c Next, use Little's law to compute system throughput.
c
                        tput(class) = N(class) / sysr(class)
80                      continue
c
c Finally, use Little's law to compute center queue lengths.
c
                    do 90 center=1,Ncents
                        qlen(n2+1,n3+1,center) = 0.0
                        do 95 class=1,3
                            if (N(class) .eq. 0) goto 95
2003                        qlen(n2+1,n3+1,center) =
     x                          qlen(n2+1,n3+1,center) +
     x                          rtime(class,center) * tput(class)
95                          continue
90                      continue
c
70                  continue
65              continue
60          continue
c
c Print results.
c
        do 100 class=1,3
            if (Ncusts(class) .eq. 0) goto 100
c
            write (6,105) class
105         format (7h Class ,i1,1h:)
            write (6,110) tput(class)
110         format (22h  System throughput: ,f8.4)
            write (6,115) Ncusts(class) / tput(class)
115         format (25h  System response time: ,f8.4)
c
            write (6,120)
120         format (24h  Device utilizations: )
            do 125 center=1,Ncents
                write (6,130) center,tput(class)*demand(class,center)
130             format (i7,2h: ,f5.3)
125             continue
```

```
c
            write (6,140)
140         format (25h   Device queue lengths: )
            do 145 center=1,Ncents
                write (6,150) center, tput(class)*rtime(class,center)
150             format (i7,2h: ,f8.4)
145             continue
c
100         continue
c
        end
```