Helsinki University of Technology
Laboratory of Telecommunications Technology

March 30, 1999

Qlib - Traffic Theory Library

Preface

****

The Qlib program library has resulted from the work conducted
over several years jointly by

Laboratory of Telecommunications Technology of
Helsinki University of Technology (HUT) and

VTT Information Technology, Telecommunications

The following individuals have contributed to the library:

Samuli Aalto, Jani Lakkakorpi, Ilkka Norros, Anssi Pirhonen
and Jorma Virtamo.

The programs can be used and modified freely.

No claims are made about the correctness of the programs
and no liability is taken for any damage caused by the use
of the programs in the library.

Reports of bugs in the program can be sent to qlib@tct.hut.fi

****

Contents

1. Installation of the library

1.1. Installation in UNIX

Traffic theory library is in a file called *traffic.tar*. Extracting (in UNIX) is executed with the following command:

tar xvf traffic.tar

Now a subdirectory called *traffic* is created in the working directory. This subdirectory contains the C-functions of the library. After this, we change the working directory to *traffic* and execute command *make*. This command will compile the library. The compiled library will be named *qlib.a*.

Manual pages for the functions are located in a subdirectory named *traffic/man/man3*. These pages are *TROFF/NROFF* -code. Manual pages can be used by *man* -program after we add subdirectory *traffic/man* to the environment variable *MANPATH*. If the shell is (t)csh, the right command is:

setenv MANPATH $MANPATH\:$HOME/traffic/man.

(It is assumed, that subdirectory *traffic* is located in the home directory.) Now we can read the manual page of a *function* by entering command:

man *function*.

Manual pages contain a lot of special characters. In order to fully exploit these manual pages, it is recommended to transform these pages into *postscript* -form before reading them. PS -versions of the manual pages are located in subdirectory *traffic/doc*. They can be easily extracted from manual pages with command:

groff -e -man *function***.3 >** *function***.ps**.

Now, with *ghostview*, manual pages can be read in their full content.

1.2. Installation in PC (with Borland C++ and Microsoft Windows)

First we extract the files from *dos.zip* to a new folder called *dos*. The source files are also included in this directory to make it possible to update the library. The library, *qlib.lib*, is rebuilt/updated (if necessary) in the following manner:

There is a project file (*qlib.ide*) in the *dos* -directory, that creates *qlib.lib*. From *Project* -menu we choose *Open project* and *qlib.ide*. Then we check, that files called *c0wl.obj*, *mathwl.lib* (both should be located in *bc4\lib*) and all our own source files are among the project files. (To add a node: click the uppermost node in the project window with the right mouse button and choose *Add node.*) From *Options/Project/Directories/Include Directories* we check the location of our header files (should be *dos*). Before building the library we choose the uppermost node in the project window, click the right mouse button and choose *TargetExpert*. The recommended options are: Target Type: *Static Library*, Platform: *Windows 3.x(16)* and Target Model: *Large*. Finally we build the library from *Project/Build all.*

2. Using the library

2.1. Use in UNIX

When we want to use the functions of this traffic theory library in some (test) program of our own (*testing.c*, for example), compiler has to know where to find *include* -files and the library. Usually *-I dir* adds the directory to the search path of *include* -files and *-L dir* to the search path of libraries. In the end of this linking command we type *'qlib.a -lm'*, where the first part is traffic theory library and the second part is math library. For example:

gcc -I../traffic -L../traffic -o testing testing.c qlib.a -lm.

2.2. Use in PC (with Borland C++ 4.0 and Microsoft Windows)

From *Project* -menu we open the desired project (for example: *testing.ide*), click the uppermost node in the project window with the right mouse button and choose *Add node*. Then we navigate to *qlib.lib* and add it to project files. To *Options/Project/Directories/Include Directories* we add the location of our header files (*dos*). Of course, we need a test program to use our library. We add this *testing.c* -file to this project and then we build the executable file from *Project/Build all*. Before building the executable program, we again choose the uppermost node (*testing.exe*) in the project window, click the right mouse button and choose *TargetExpert*. The recommended options are: Target Type: *EasyWin*, Platform: *Windows 3.x(16)* and Target Model: *Large*.

2.3 Mathematica packages

2.3.1 UNIX

Mathematica -versions of functions of the traffic theory library are located in a file called *Qlib.m.* (If the *Mathematica* -version of that particular function exist.) In UNIX, we start *Mathematica* with commands:

use math or **use mathematica**
math or **mathematica**.

Then we take our package into use with command:

Get["Qlib`"].

2.3.2 PC

In PC environment we can use the same *Mathematica* -functions as in UNIX. After *Mathematica* is started, we change the working directory to (for example) *packages*:

SetDirectory["c:\full_path\packages"]

Then we take the package into use with command:

Get["Qlib`"].

NOTE: These Mathematica -versions are usually much slower than the C-functions.

2.4. MathLink

2.4.1 UNIX

tar -files *mlunix.tar* and *mlunixhelp.tar*.can be extracted in UNIX just like the file *traffic.tar* in section 1.1.

After extracting *mlunix.tar*, we have a directory called *link*. There we have a program called *qlib*, which includes all the functions of the traffic theory library except the *ams* -functions. These functions have the same names as the *Mathematica* -functions added with a Lnk - prefix to make these concepts separate. *Makefile*, source code and the template files are in this directory, too.

qlib -program can be recompiled with command *make* (see *Makefile* first!) if a *mcc* -compiler for *MathLink* is in use. (If this is not the case, extract *mlunixhelp.tar* and see Todd Gayley's *MathLink Tutorial.*) *qlib* needs a compiled *qlib.a* -library in the same directory. (It can be copied for example from *traffic* -directory.)

For using *MathLink* there are at least two good sources of information: Todd Gayley's *A MathLink Tutorial* and the chapter 2.12 from the *Mathematica -manual.* Here is a short example of how to use a C-function of the traffic theory library from *Mathematica:*

- Start Mathematica: **use math**
  math
- Install the library: In[1] := **Install["qlib"]**
  Out[1] = LinkObject['./qlib', 1, 1]
- Use function Qmd1: In[2] := **LnkQmd1[3, 0.4]**
  Out[2]= 0.00458191
- Quit using library : In[3] := **Uninstall["qlib"]**
  Out[3]= qlib

2.4.2 PC (with Borland C++ and Microsoft Windows)

For *Borland 4.0* there is a project file *lnk.ide* in *mldos* -directory (first extract *mldos.zip* into a new folder called *mldos*), that creates *qlib.exe*. The parts of the project are *qlib.lib*, Mathematica's *mlink16.lib*, *template.def*, *qlib.c*, and *qlibtm.c*..

It seems that the DOS -version of *MathLink* disconnects, if the return value of a real function is zero.

Here is a short (and hopefully clear) version of what you have to do to get your *qlib.exe* running. (You can first try to just run *qlib.exe*. If it doesn't work, you may have to rebuild it.):

- Get all the necessary tools (*mprep* etc.): *MathLink for Windows Developer's Kit* can be downloaded from *MathSource (www.mathsource.com). winmldk.zip* is included in *mldos* -directory, too.

- Preprocess *qlib.tm* into *qlibtm.c*
  Type the following command at the DOS prompt:
  ```
  mprep qlib.tm -o qlibtm.c
  ```

- Create a new project file (or modify *lnk.ide*) for a Windows application
  Launch *Borland C++ 4.0*.
  Choose *New* from the *Project* menu.
  In the *New Project* dialog box:
  Target Type: "*EasyWin*"
  Platform: "*Windows 3.x (16)*"
  Target Model: "*Large*"
  Click the OK button to close the *New Project* dialog box.

- Add the source files *qlib.c, qlibtm.c, template.def, qlib.lib* and *mlink16.lib*
  In the Project -window that appears next:
  Add the files (first click the right mouse button on the uppermost Project item (*.*exe*)): *qlib.c, qlibtm.c, template.def, qlib.lib and mlink16.lib.*

- Set project options
  Choose *Project* from the *Options* menu.
  In the *Project Options* dialog box:
  Add *D:\wnmath22\mathlink\include* (*mathlink.h*) and *C:\full_path\dos* (your own header files) to the compiler include path. They are separated with a semicolon (;).
  Click the OK button.

- Build the project
  Choose *Build All* from the *Project* menu.

- Run the executable file
  Click the *qlib.exe* -file.
  In the text box labeled MathLink, type `qlib and` click the OK button.

- `Install["qlib", LinkMode->Connect]` in *Mathematica*
  Launch *Mathematica*

Evaluate the following expressions:

```
Install["qlib", LinkMode->Connect]
LnkQmd1[3,0.4]
Uninstall["qlib"]
```

- Another way to run *qlib.exe* in *Mathematica* is to evaluate the following expression:

```
Install["C:\full_path\qlib "]
```

2.4.3. TCP/IP -connections

Mathlink's DLL -libraries demand *WINSOCK.DLL* for TCP/IP -connections. It can be obtained with the following combination:

- *Microsoft Windows for Workgroups 3.11*
- *Microsoft TCP/IP-32*
- *Microsoft Win32s 1.15 or newer*

TCP/IP -connections are used with the following options:

Command line:

-mathlink
-linkname 12345 (IP port number)
-linkmode listen
-linkprotocol tcp

Mathematica:

"12345"
LinkMode->Connect
LinkProtocol->"TCP"
LinkHost->"hostname"

- **An example of how to use qlib in a UNIX -computer from your own PC:**

In alpha.hut.fi:

qlib -mathlink -linkname 12345 -linkmode listen -linkprotocol tcp

In Mathematica (PC):

Install["12345", LinkMode->Connect, LinkProtocol->"TCP",
LinkHost->"alpha.hut.fi"]

- **An example of how to run a remote kernel in a UNIX -computer from your own PC:**

In alpha.hut.fi:

math -mathlink -linkname 12345 -linkmode listen -linkprotocol tcp

In Mathematica (PC):

Choose *Options/Kernels* and create new *Specific Kernel*:

Description: alpha
Link Protocol: TCP
Link Mode: Connect
Link Name: 12345
Link Host: alpha.hut.fi

Then choose *Connect to Kernel.*

Using remote kernel in alpha will increase notably the speed of computing.

NOTE: When you enter the command

math -mathlink -linkname 12345 -linkmode listen -linkprotocol tcp

in a UNIX -computer, *Mathematica* starts listening and we all can contact it with our *Front Ends* (before you contact it) if we only know the name of the link (here "12345"). So, the link name should be considered as a password and it should not be told to anyone.

2.5 Traffic theory library function list

Function name and call pattern (MathLink -functions) (1)	**Function name in Mathematica-package (⩾ same call pattern)**	**Description**
-	**CtMarkovChain** [**Matrix**]	Gives the stationary probabilities of a continuous time Markov chain with the transition rate matrix Q.
-	**DtMarkovChain** [**Matrix**]	Gives the stationary probabilities of a discrete time Markov chain with the transition probability matrix P.
-	**MVA** [**Matrix, Vector, Integer**]	Algorithm for the Mean Value Analysis of a closed Jackson network. It returns the average queue lengths and the average sojourn times in the queues. The branching ratios are given by the matrix R; the vector mu specifies the service rates of the queues; and K is the number of customers in the network.
LnkBerli[Integer, Real] (2)	**Berli**	Erlang loss probability.
LnkBerld[Real, Real] (2)	**Berld**	Erlang loss probability.
LnkXerl[Real, Real]	**Xerl**	Inverse Erlang function.
LnkAerl[Real, Real]	**Aerl**	Inverse Erlang function.
LnkBkaufman[Integer, Integer, Integer, IntegerList, RealList]	-	Erlang blocking probability for multiple traffic classes.
LnkBmitra[Integer, Integer, Real, IntegerList, RealList]	-	Erlang blocking probability for multiple traffic classes.
GAMS (3)	-	Anick-Mitra-Sondhi handling function.
EvAMS (3)	-	Anick-Mitra-Sondhi handling function.
initAMS (3)	-	Anick-Mitra-Sondhi handling function.
freeAMS (3)	-	Anick-Mitra-Sondhi handling function.
LnkQmd1[Real, Real]	**Qmd1**	Virtual waiting time distribution for the M/D/1 queue.

LnkQndd1[Real, Integer, Real]	**Qndd1**	Virtual waiting time distribution for the N*D/D/1 queue..
LnkQsdd1[Real, RealList]	**Qsdd1**	Virtual waiting time distribution for the $\sum D_i / D / 1$ queue.
LnkMg1[Integer, Real, Integer]	**Mg1 (4)**	Queue length probability function for the M/G/1 queue.
LnkQmxd[Real, Real, RealList]	**Qmxd1**	Unfinished work tail distribution function for the $M^x / D / 1$ queue.
LnkFmd1[Real, Real]	**Fmd1**	Virtual waiting time distribution for the M/D/1 queue.
LnkIntFmd1[Integer, Real]	**IntFmd1**	Virtual waiting time distribution for integral values of the amount of unfinished work in the system..
LnkSumMd1[Real, Real]	**SumMd1**	Calculates state probabilities of the M/D/1 queue.
LnkRecMd1[Integer, Real]	**RecMd1**	Calculates state probabilities of the queue with a recursive algorithm.
LnkFmdn[Real, Real, Integer]	**Fmdn**	Virtual waiting time distribution of the M/D/n queue.
LnkIntFmdn[Integer, Real, Integer]	**IntFmdn**	Virtual waiting time distribution for integral values of the amount of unfinished work in the system..
LnkMdn[Real, Real, Integer]	**Mdn**	Calculates state probabilities of the M/D/n queue.
LnkFekdn[Real, Real, Integer, Integer]	**Fekdn**	Virtual waiting time distribution for the $E_k / D / n$ queue.
LnkFend1[Real, Integer, Real]	**Fend1**	Virtual waiting time distribution for the $E_n / D / 1$ queue.

(1) These are the *MathLink* function names. The C-functions lack the Lnk- prefix and their call pattern is different. See the manual pages for more information.

(2) The C-functions are called *Berl_i* and *Berl_d*.

(3) Anick-Mitra-Sondhi handling functions are not included in the *qlib*-file and cannot be used via *MathLink*. No *Mathematica* -versions of these functions are either available.

(4) Call pattern for *Mathematica* -version of *Mg1* is **Mg1[Integer, Real, Function].** Here is an example - first we define function F: **F[t_] := If[t < 1, 0, 1]** (M/D/1) and then use *Mg1*: **Mg1[3, 0.4, F].**

See manual pages of the functions for more information.

3. Maintenance of the library

3.1 Maintenance in UNIX

When we want to add a new file to the library, we add the name of the file to *OBJS* -line of *Makefile* (we replace '.c' with '.o'). In the end of *Makefile* are the dependencies of this file from other files. An example: if the added file is called *func.c* and it has two header -files, *func1.h* and *func2.h*, we add the dependency in a following way:

func.c: func1.h func2.h.

After this, we compile the library by command *make*. Now the tar -file can be created with the command:

tar cvf traffic.tar traffic/

in the parent directory of *traffic*.

Library is compiled with optimizations. If we want to debug the library, the '-O9' in line *CFLAGS* of *Makefile* has to be replaced with '-g'.

3.2 Maintenance in PC

First we open the project file of the library (*qlib.ide),* from *Project* -menu. Then we click the uppermost node in the project window with the right mouse button, add the desired files, add the locations of the possible new header files to *Include Directories* and rebuild the library (Section 1.2).

qlibtm.c -file has to be updated (from *qlib.tm* with *mprep*) and *qlib.exe* -file has to be rebuilt (Section 2.4.2), if we want to use our new function(s) from *Mathematica*..