

# The qnetworks Toolbox

A Software Package for Queueing Networks Analysis

Moreno Marzolla  
marzolla@cs.unibo.it  
<http://www.moreno.marzolla.name/>

Università di Bologna

april 28, 2010

## Talk Outline

Introduction

A bird-eye view of GNU Octave

The qnetworks Toolbox  
Single-station Queueing Systems  
Queueing Networks

Comparison with JMT

Conclusions

References

# What is qnetworks?

qnetworks is a package for Queueing Networks analysis.

- ▶ Written in GNU Octave, an interpreted language for numerical computations
- ▶ qnetworks implements the most important algorithms for QN analysis (MVA, convolution, Jackson networks...);
- ▶ qnetworks also provides some numerical algorithms for Markov Chains analysis (computation of steady-state residency probabilities, mean time to absorption, time-averaged sojourn time...);
- ▶ Open source software (GNU GPL version 3)

<http://www.moreno.marzolla.name/software/qnetworks>

# Yet Another QN Package?

- ▶ Over the years, quite a lot of QN packages have been developed (see <http://web2.uwindsor.ca/math/hlynka/qsoft.html> for a list)
- ▶ Unfortunately, most of those packages have vanished; the surviving ones are of very limited scope, obsolete and/or unmaintained

# GNU Octave

- ▶ GNU Octave is a free Matlab® clone. Octave/Matlab are widely used for implementing numerical algorithms, due to their very compact notation.
- ▶ `qnetworks` is entirely written as m-scripts which run inside the Octave interpreter.
- ▶ This allows maximum portability, as the Octave interpreter runs on the most common platforms (Windows, MacOSX, Linux and most UNIX variants).
- ▶ The execution speed of the interpreter is not a limiting factor for `qnetworks` (see later).

## GNU Octave

### Example

Solve  $Ax = b^T$  where  $A \leftarrow \begin{pmatrix} 12 & 8 & 7 \\ 21 & -9 & 8 \\ 3 & 3 & 1 \end{pmatrix}$  and  $b \leftarrow (0.75 \ 1 \ 3)$ .

```
octave> A = [12 8 7; 21 -9 8; 3 3 1];
```

```
octave> b = [0.75 1 3];
```

```
octave> A\b'
```

```
ans =
```

```
1.43120
```

```
0.56686
```

```
-2.99419
```

# GNU Octave

## Vector manipulation

GNU Octave provides useful features for manipulating vectors

```
octave> a = [1 7 6 1 3 8 2 9 4 2]
a =
    1    7    6    1    3    8    2    9    4    2
octave> sum(a)
ans = 43
octave> a>3
ans =
    0    1    1    0    0    1    0    1    1    0
octave> a(a>3)
ans =
    7    6    8    9    4
```

# Octave

## MVA algorithm

### MVA algorithm

```
for k ← 1 to K do
  Qk ← 0
endfor
for n ← 1 to N do
  for k ← 1 to K do
    if k is a delay center then
      Rk ← Dk
    else
      Rk ← Dk(1 + Qk)
    endif
  endfor
  X ← n / (Z + ∑k=1K Rk)
  for k ← 1 to K do
    Qk ← XRk
  endfor
endfor
```

### Octave implementation

```
## dd(k) == 1 iff server k is delay center
function [R,Q] = mva(N, V, D, Z, dd)
  Q = zeros(size(V));
  for n=1:N
    R = D .* (1 + Q.*(dd == 0));
    X = n / (Z+sum(R));
    Q = X*R;
  endfor
endfunction
```

## Note on notation

What does the expression `D .* (1 + Q .* (dd == 0))` compute?

Let:

```
D = [15 3 6];
Q = [1 7.4 2];
dd = [1 0 0];
```

```
dd==0 Q .* (dd==0) 1 + Q.*(dd == 0) D .* (1 + Q.*(dd == 0))
```

- ▶ is the boolean vector `[0 1 1]`
- ▶ is the element by element product of `[1 7.4 2]` and `[0 1 1]` ( $\rightarrow$  `[0 7.4 2]`)
- ▶ adds 1 to all elements of `[0 7.4 2]` ( $\rightarrow$  `[1 8.4 3]`)
- ▶ is the element by element product of `[15 3 6]` and `[1 8.4 3]` ( $\rightarrow$  `[15 25.2 18]`)

## Note on notation

Thus, the expression

```
R = D .* (1 + Q.*(dd == 0))
```

is equivalent to this pseudo code:

```
for k ← 1 to K do
  if k is a delay center then
     $R_k \leftarrow D_k$ 
  else
     $R_k \leftarrow D_k(1 + Q_k)$ 
  endif
endfor
```

# The qnetworks Toolbox

## Single-station Queueing Systems

qnetworks computes utilization, response time, average number of requests and throughput of several kinds of single-station queueing systems:

- ▶  $M/M/1$ ,  $M/M/m$
- ▶  $M/M/1/k$ ,  $M/M/m/k$
- ▶  $M/M/\infty$
- ▶ asymmetric  $M/M/m$  which contains  $m$  servers with possibly different service rates
- ▶  $M/G/1$  (general service time distribution)
- ▶  $M/H_m/1$  (hyperexponential service time distribution)

## Example

### Single Station Queueing Systems

Let us consider a  $M/M/1$  system

- ▶ Arrival rate  $\lambda = 0.3$  jobs/second
- ▶ Service rate  $\mu = 0.4$  jobs/sec

We can compute the utilization  $U$ , response time  $R$ , average number of requests in the system  $Q$  and throughput  $X$  as follows:

```
octave> lambda = 0.3;
octave> mu = 0.4;
octave> [U R Q X] = qnmm1(lambda, mu)
U = 0.75000
R = 10.0000
Q = 3.0000
X = 0.30000
```

## Example

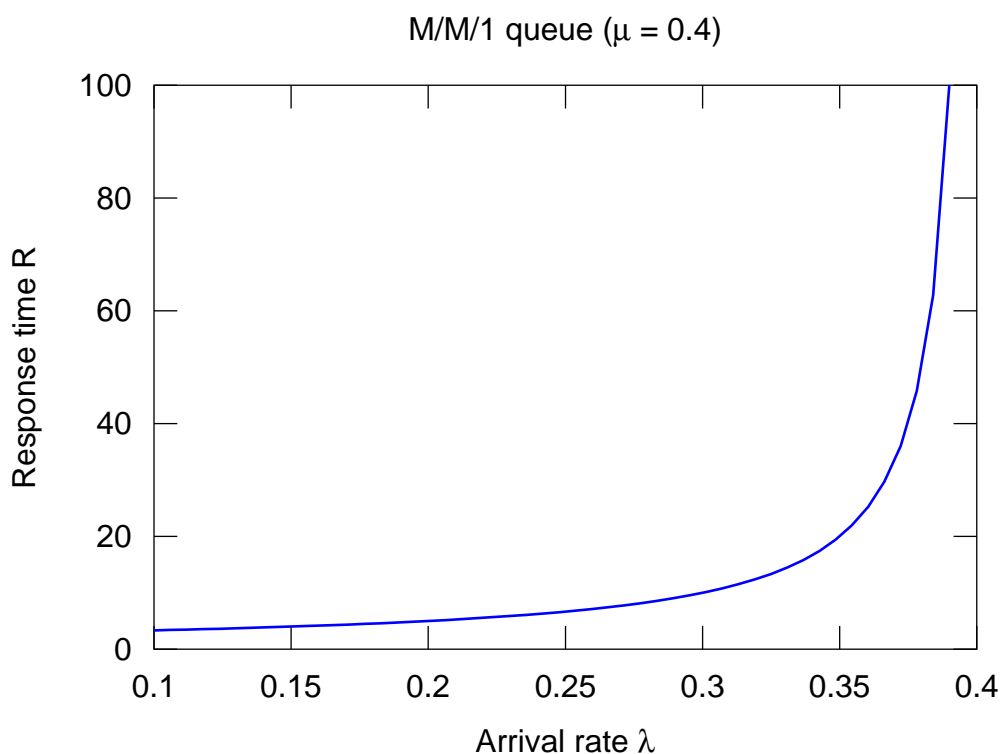
### Single Station Queueing Systems

Let us examine how the response time  $R$  grows as the arrival rate  $\lambda$  approaches the service rate  $\mu = 0.4$

```
mu = 0.4;
lambda = linspace(0.1,0.39,50);
R = zeros(size(lambda));
for i=1:length(lambda)
    [nc R(i)]=qnmm1(lambda(i),mu);
endfor
plot(lambda,R);
```

## Example

### Single Station Queueing Systems



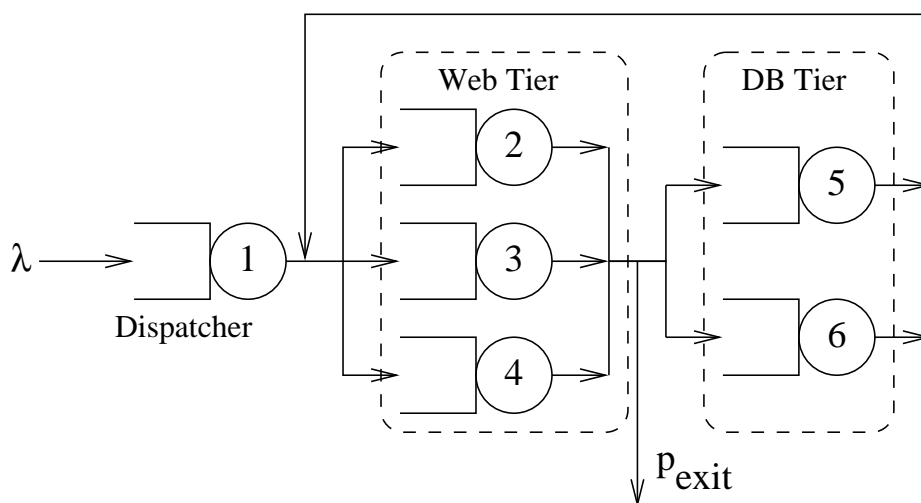
# Queueing Networks

Function Name	Supported network type			
	Open	Closed	Single	Multi
qnopensingle()	✓	-	✓	-
qnopensmulti()	✓	-	-	✓
qnconvolution()	-	✓	✓	-
qnconvolutionld()	-	✓	✓	-
qnclosedsinglemva()	-	✓	✓	-
qnclosedsinglemvald()	-	✓	✓	-
qnclosedmultimva()	-	✓	-	✓
qnclosedmultimvaapprox()	-	✓	-	✓
qnmix()	✓	✓	-	✓
qnsolve()	✓	✓	✓	✓
qnmvablo()	-	✓	✓	-
qnmakov()	-	✓	✓	-
qnopensab()	✓	-	✓	-
qnclosedab()	-	✓	✓	-
qnopensbsb()	✓	-	✓	-
qnclosedbsb()	-	✓	✓	-
qnclosedgb()	-	✓	✓	-

# Queueing Networks

We illustrate qnetworks capabilities for solving QNs using a simple example.

We consider a three-tier E-commerce site

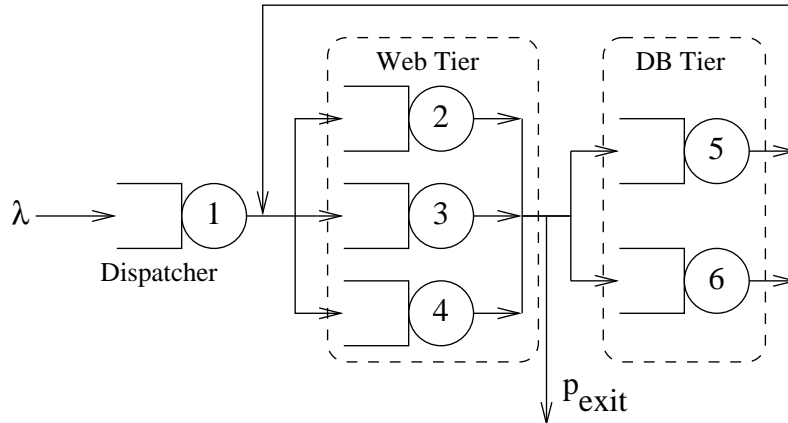


- ▶ Dispatcher (center 1, service time  $S_1 = 0.5$ );
- ▶ Web Servers (centers 2–4, service time  $S_2 = S_3 = S_4 = 0.8$ );
- ▶ Database Servers (centers 5–6, service time  $S_5 = S_6 = 1.8$ );



# E-Commerce example

Routing probability matrix

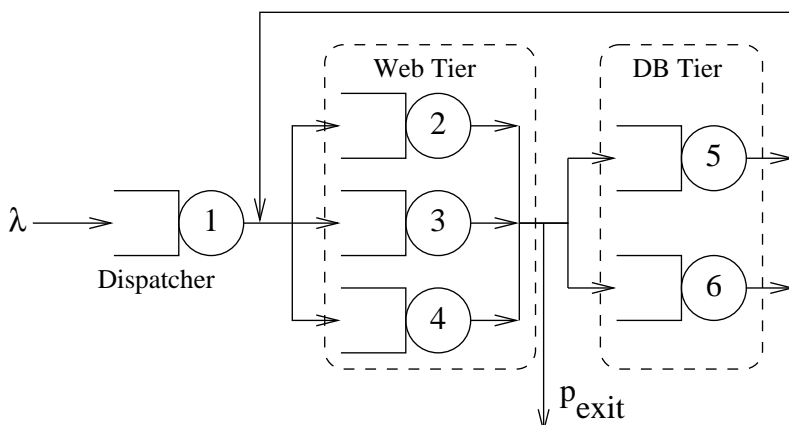


Assuming  $p_{\text{exit}} = 0.5$  and uniform routing, the routing probability matrix  $P$  is

$$P = \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 0 & 0 & 1/4 & 1/4 \\ 0 & 1/3 & 1/3 & 1/3 & 0 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 & 0 \end{pmatrix}$$

# E-Commerce example

Building the model



```
p_exit = 0.5; # exit probability
i = 2:4; # indexes of Web servers
j = 5:6; # indexes of DB servers
P = zeros(6,6);
P(1,i) = 1/3;
P(i,j) = (1-p_exit)/2;
P(j,i) = 1/3;
S = [0.5 0.8 0.8 0.8 1.8 1.8];
lambda = [0.1 0 0 0 0 0];
V = qnvisits(P,lambda);

[U R Q X] = \
    qnopensingle(sum(lambda),S,V);
```

## E-commerce example

### Results

The statement  $V = \text{qnvisits}(P, \lambda)$  computes the visit counts  $V_k$ , which satisfy:

$$V_k = \lambda_k + \sum_{j=1}^K V_j P_{jk}$$

In the previous example we have

$$V_1 = 1, V_2 = V_3 = V_4 = 0.6\bar{6} \text{ and } V_5 = V_6 = 0.5$$

and we obtain the following performance results:

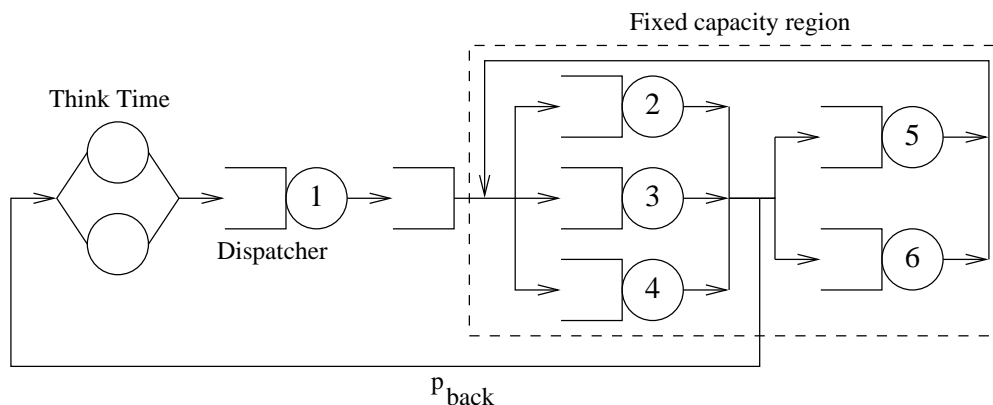
$$\begin{aligned} U &= (0.050000, 0.053333, 0.053333, 0.053333, 0.090000, 0.090000) \\ R &= (0.526320, 0.845070, 0.845070, 0.845070, 1.978020, 1.978020) \\ Q &= (0.052632, 0.056338, 0.056338, 0.056338, 0.098901, 0.098901) \\ X &= (0.100000, 0.066667, 0.066667, 0.066667, 0.050000, 0.050000) \end{aligned}$$

## E-commerce example

### Flow-Equivalent Centers

Let us consider a slightly more sophisticated example.

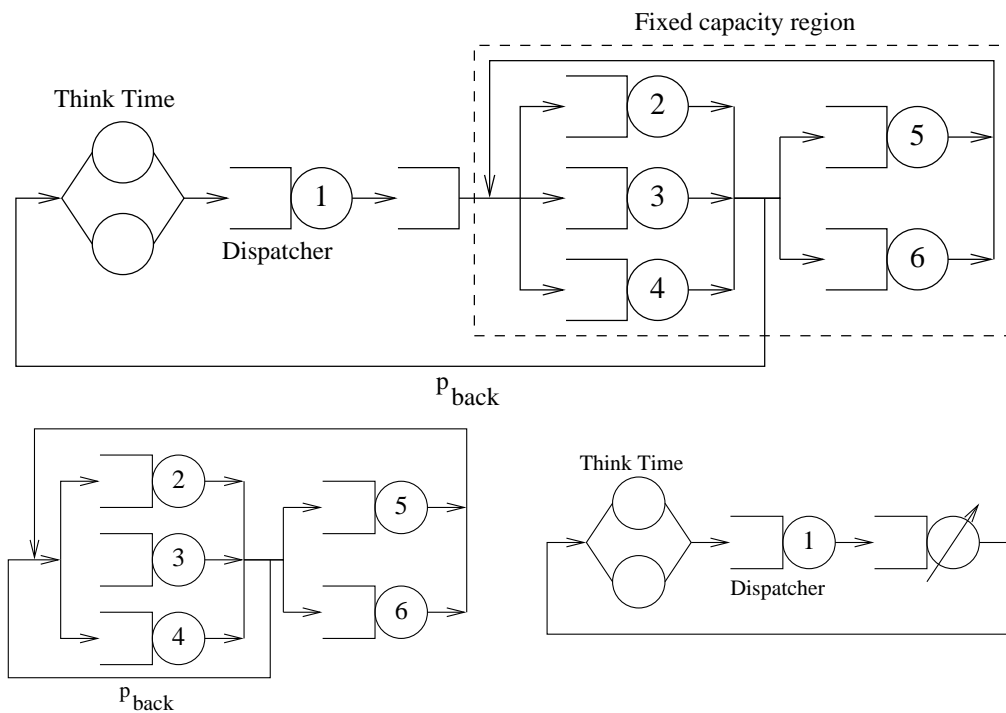
- ▶ The system is modeled as a closed network with  $N$  customers.
- ▶ There is a capacity constraint of  $M$  requests behind the dispatcher



## E-commerce example

### Flow-Equivalent Centers

We replace the fixed capacity region with a load-dependent service center and solve the resulting model (which does have product form solution).



## Analyzing the capacity-constrained model

We start by defining the complete model

```

p_back = 0.5;      # back probability
i = 2:4;          # range of Web servers
j = 5:6;          # range of DB servers
P = zeros(6,6);
P(1,i) = 1/3;
P(i,j) = (1-p_back)/2;
P(i,1) = p_back;
P(j,i) = 1/3;
S = [0.5 0.8 0.8 0.8 1.8 1.8];
V = qnvisits(P); # Compute visit counts
Z = 5;           # Think Time
N = 20;          # Population
m = ones(1,6);  # m(k)=number of servers at center k

```

Then we short-circuit the capacity constrained region by setting all other service times to zero

```
S(1) = 0
```

## Analyzing the capacity-constrained model

We solve the short-circuited submodel by computing the throughput  $X_{\text{sub}}(n)$  along the removed node(s) as a function of the population size  $n = 1, 2, \dots, M$ :

$$S_{\text{sub}}(n) = \begin{cases} 1/X_{\text{sub}}(n) & \text{if } 1 \leq n \leq M \\ 1/X_{\text{sub}}(M) & \text{if } M < n \leq N \end{cases}$$

This can be done with the following code:

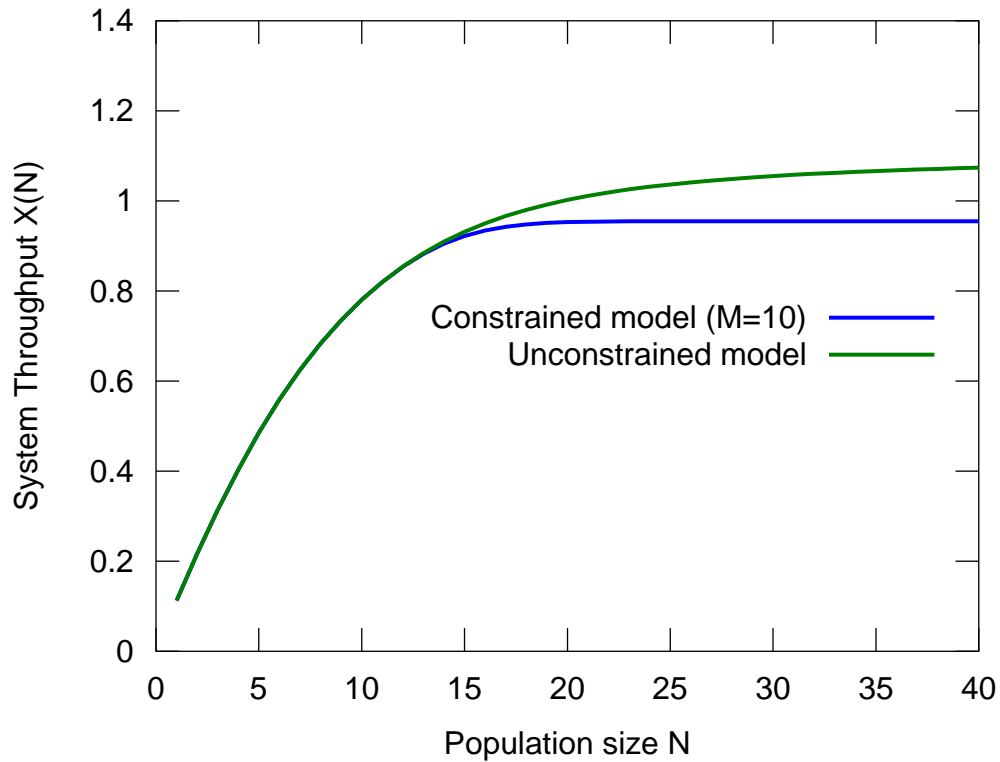
```
Ssub = zeros(1,N); # Initialize to zero
M = 10;           # Capacity constraint
for n=1:M
    [U R Q X] = qnclosedsinglemv(n,S,V);
    Ssub(n) = V(1)/X(1);
endfor
Ssub(M+1:N) = Ssub(M);
```

## Analyzing the capacity-constrained model

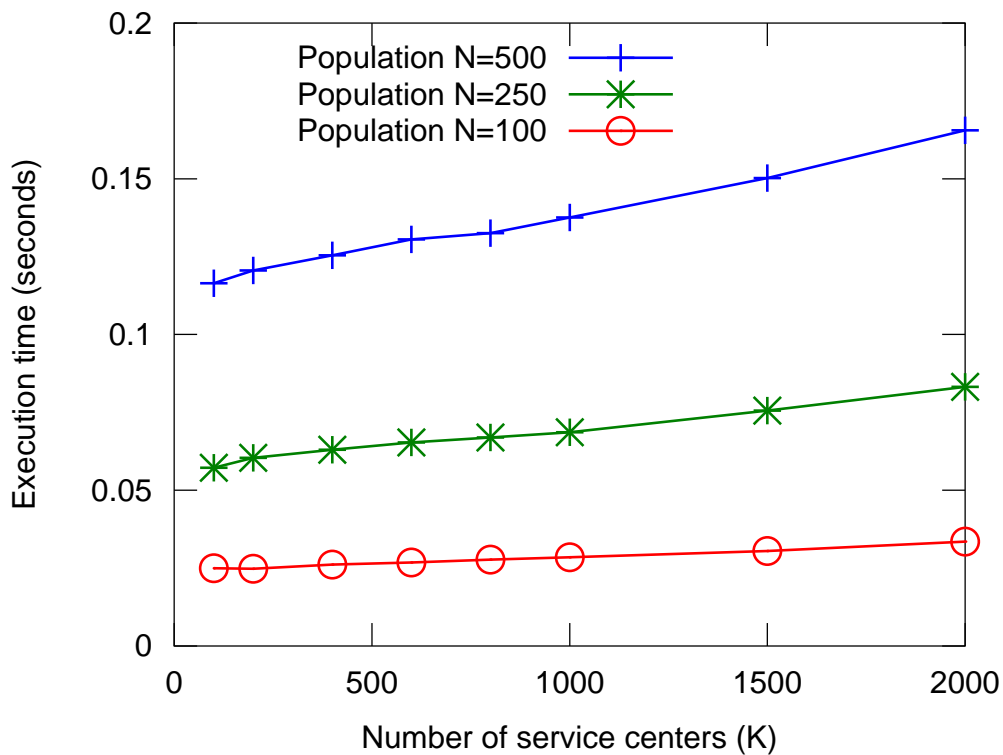
We can now solve the equivalent model (with the capacity constrained region replaced by the FESC)

```
S = [ 0.5*ones(1,N); Ssub ];
V = [1 1];
Z = 5;
[U R Q X] = qnclosedsinglemvld(N,S,V,Z);
```

## Analyzing the capacity-constrained model



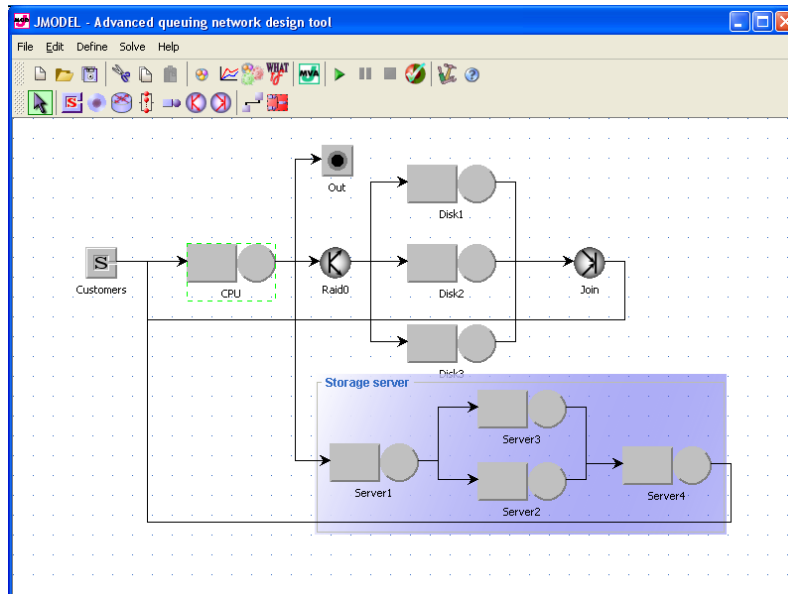
## Performance Considerations



Execution times of the `qnclosedsinglemva` function on a Linux PC, AMD Athlon 64 X2 Dual Core processor 3800+ with 2GB of RAM. Each data point is the average execution time of 5 runs.

# Java Modelling Tools (JMT)

JMT is a nice alternative to qnetworks. JMT is written in Java and is available at <http://jmt.sourceforge.net/>



## JMT vs qnetworks

	JMT	qnetworks
Solution technique	Simulation	Analytical
Graphical user interface	Yes	No
Allows to programmatically build and solve models?	No	Yes
Supports Product-Form networks?	Yes	Yes
Supports Non PF networks?	Yes	Some
Free and open source?	Yes (GPLv2)	Yes (GPLv3)

# Conclusions

qnetworks is an open source QN analysis package for GNU Octave. It supports:

- ▶ single station queueing systems;
- ▶ open, closed or mixed networks with multiple job classes;
- ▶ the MVA (single and multi class) and convolution (single class only) algorithm for product-form networks;
- ▶ computation of performance bounds (Asymptotic Bounds, Balanced System Bounds, Geometric Bounds);
- ▶ approximate MVA for closed QN with blocking;
- ▶ Markov Chains analysis

Many new algorithms still to be implemented. Would you like to help?

<http://www.moreno.marzolla.name/software/qnetworks/>

# References



Marco Bertoli, Giuliano Casale, and Giuseppe Serazzi.  
JMT: performance engineering tools for system modeling.  
*SIGMETRICS Perform. Eval. Rev.*, 36(4):10–15, 2009.



G. Bolch, S. Greiner, H. de Meer, and K. Trivedi.  
*Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*.  
Wiley, 1998.



Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik.  
*Quantitative System Performance: Computer System Analysis Using Queueing Network Models*.  
Prentice Hall, 1984.



M. Reiser and S. S. Lavenberg.  
Mean-value analysis of closed multichain queueing networks.  
*Journal of the ACM*, 27(2):313–322, April 1980.