

Automated multi-paradigm analysis of
extended and layered queueing
models with LINE

Giuliano Casale

Department of Computing
Imperial College London
g.casale@imperial.ac.uk

LINE Solver (line-solver.sf.net)

- MATLAB library for system performance and reliability analysis based on queueing theory
- Ver 2.0.0: Major tool overhaul and refactoring

Skip to: [Videos](#) | [Downloads](#) | [Resources](#)

LINE

Performance and Reliability Analysis Engine

[Home](#)

[Downloads](#)

[Manual](#)

[Wiki](#)

[API](#)

[Videos](#)

[Resources](#)

Support

[Help forum](#)

[Report a bug](#)

[Request a feature](#)

[Sourceforge site](#)

What is LINE?

LINE is an open source MATLAB library for system performance and reliability analysis based on queueing theory.

Main features

The tool offers a language to specify **extended queueing networks** and **layered queueing networks** together with analytical and simulation-based techniques for their solution.

Models are solved in LINE with either native algorithms (CTMC, fluid, simulation, MVA, ...) or via external solvers, such as **JMT**, **LQNS**, and **BuTools**. The tool output metrics include throughputs, utilizations, response times, queue-lengths, and state probabilities. Metrics can be averages or distribution/percentiles, either in steady-state or transient regime.

Download

Download the **latest release** for MATLAB (version 2018a or later) or clone the **source code** repository. Installation information is available in the **README** file.

What's new

- I. Object-oriented language to model extended and layered queueing networks (EQNs / LQNs)
- II. Model specification fully decoupled from analysis paradigm
- III. Seamless integration with JMT, LQNS, BuTools solvers

Line in numbers:

- 40+ algorithms
- 100+ lang. classes
- 13 types of analyses
- 14 node types
- 13 sched./routing strategies
- 4 metrics

JMT Integration

Seamless two-way integration:

- Define a model in MATLAB, visualize it in JMT
- Define a model in JMT, auto-generate LINE script

MATLAB R2018a - academic use

HOME PLOTS APPS

addpath line.s

C:\Users\csg\Dropbox\code\line-dev.git\examples

```
>> model = Network('M/M/1');
source = Source(model, 'mySource');
queue = QueueingStation(model, 'myQueue', SchedStrategy.FCFS);
sink = Sink(model, 'mySink');
oclass = OpenClass(model, 'myClass');
source.setArrival(oclass, Exp(1));
queue.setService(oclass, Exp(2));
model.link(Network.serialRouting(source,queue,sink));
SolverJMT(model).getAvgTable
ans =
2x6 table
   Station   Class   QLen   Util   RespT   Tput
   _____   _____   _____   _____   _____   _____
   'mySource'   'myClass'         0         0         0   0.99694
   'myQueue'   'myClass'   0.90717   0.49595   0.98731   0.99547
```

```
>> model.jsimView
JMT Model: C:\Users\csg\AppData\Local\Temp\jsimg\tp187cd49a_9e52_4d4a_9b55_a4c0dc0a0bf8.jsimg
```

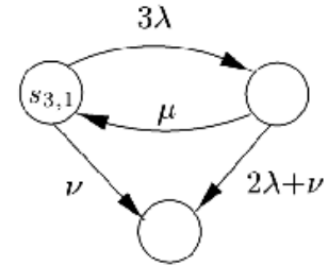
JMSIMgraph - Graphical Queueing Netw...

File Edit Define Solve Help

EQN Analysis

Continuous-Time
Markov Chains

`SolverCTMC(model).getAvg`



Fluid ODEs

`SolverFluid(model).getAvg`

$$\frac{d\mathbf{x}(t)}{dt} = F(\mathbf{x}(t)), t \geq 0,$$

Java Modelling
Tools

`SolverJMT(model).getAvg`



Mean-Value
Analysis

`SolverMVA(model).getAvg`

$$Q_i(N-1) = \frac{N-1}{N} Q_i(N)$$

EQN Analysis

Matrix-Analytic
Methods

SolverMAM(**model**).getAvg

$$P = \begin{pmatrix} B_0 & B_1 & B_2 & B_3 & \cdots \\ A_0 & A_1 & A_2 & A_3 & \cdots \\ & A_0 & A_1 & A_2 & \cdots \\ & & A_0 & A_1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

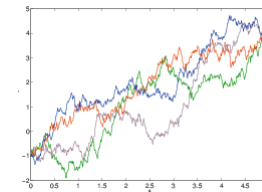
Normalizing
Constant

SolverNC(**model**).getAvg

$$Q_i(N-1) = \frac{N-1}{N} Q_i(N)$$

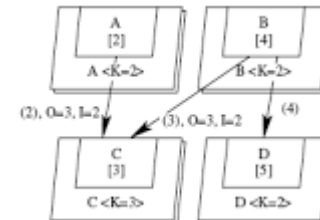
Stochastic
Simulation

SolverSSA(**model**).getAvg



LQNS/LQSIM

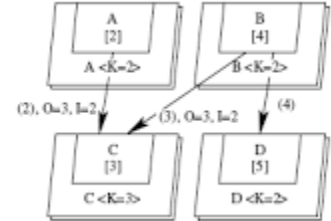
SolverLQNS(**lqnmodel**).getAvg



Parametric Solvers (LQN, Rand. Env.)

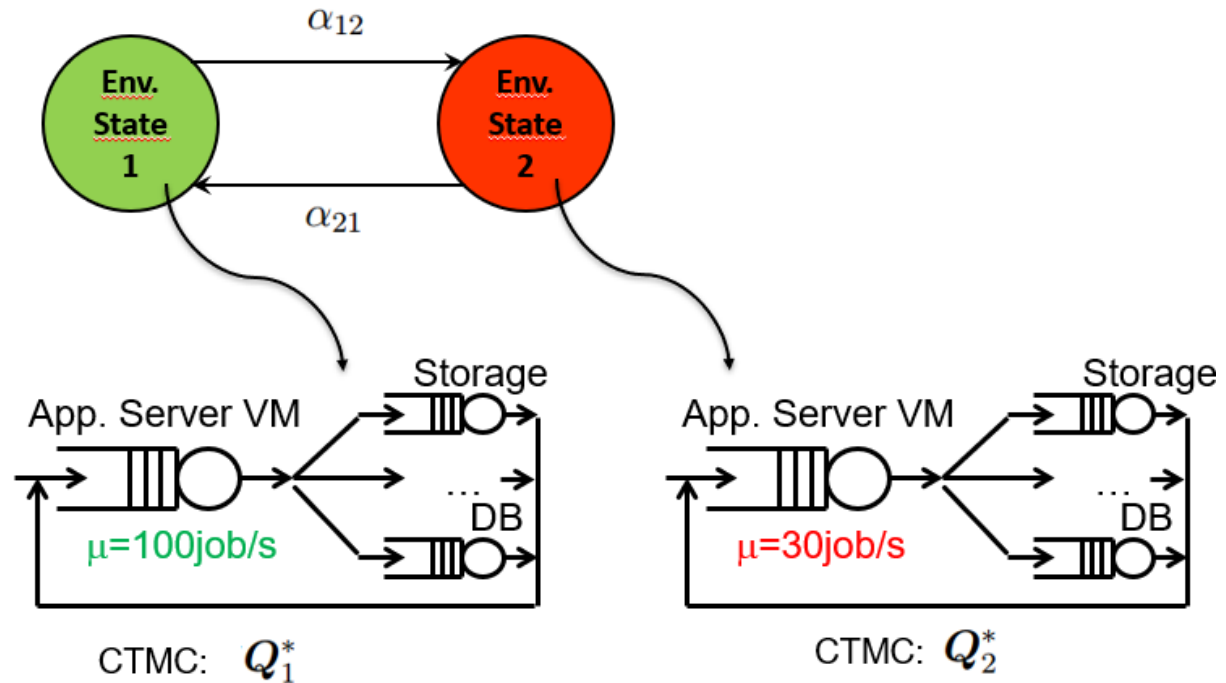
Line
Layered Solver

SolverLN(model, @(layer) SolverJMT(layer))



SolverEnv(models, env, @(submodel) SolverFluid(submodel))

Random
Environments



Demos

1. A M/M/1 queue
2. A multiclass M/G/1 queue
3. Machine interference problem
4. Round-robin load balancing
5. Modelling a re-entrant line
6. A queueing network with caching
7. Response time distribution and percentiles
8. Optimizing a performance metric