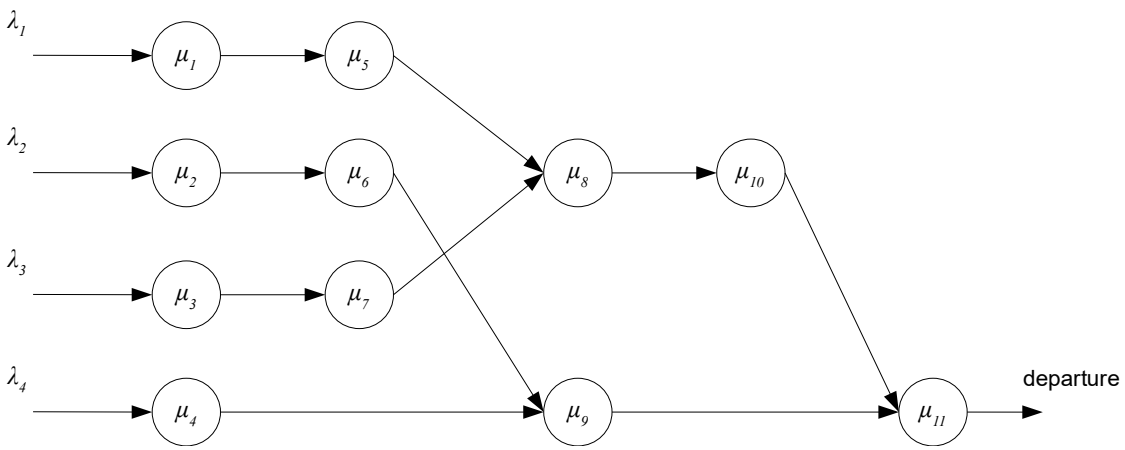
Exercise 4 – Queueing networks and Simulation-based optimization

4.1

Consider the following queueing network. Nodes represent servers and arrows the movement of customers between the servers. Customers arrive from 4 different sources and go through the system of 11 servers. The service capabilities at all servers equal 1, queue lengths are unlimited and queueing disciplines FIFO. Inter-arrival times and service times are all exponentially distributed with means λ_i, i={1,2,3,4} and μ_i, i={1,2,...,11}.



Build a simulation model of the system to estimate the average total time in system for the first 100 customers (that have departed). System starts empty end idle.

Take:

$$\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}=\{0.5, 1, 1, 0.5\}$$

$$\{\mu_1, \mu_2,..., \mu_{11}\}=\{0.4, 0.7, 0.5, 0.3, 0.3, 0.8, 0.2, 0.5, 0.4, 0.5, 0.2\}$$

Now, examine Matlab's functionality for creating response surface models (under the help for Statistics toolbox). Devise and simulate a central composite design to determine a full quadratic regression metamodel for the queueing network with respect to mean processing times. You may decrease and increase the above parameters for μ_i with 10% to form the – and + levels for the factors in the design.

Since the simulation responses for the factor combinations have different variances, use weighted least squares estimates for the parameters of the regression model. The estimates are calculated through

$$\hat{\beta}=(X'cov(Y)^{-1}X)^{-1}X'cov(Y)^{-1}Y$$

where *X* denotes the design matrix and the covariance matrix of the simulation responses *Y* is a diagonal matrix where the non-zero element in the i^{th} row is the variance estimate of the responses corresponding to the i^{th} factor combination.

To form confidence intervals for the regression coefficients, calculate jackknifing observations of the coefficients by recalculating the estimates with one of the simulation replications deleted. If *r* replication were initially performed for the design, *r* jackknifing observations are obtained. These observations are

$$\hat{P}_i = r\hat{\beta}_i - (r-1)\beta_i^{(-k)}$$

where $\beta_i^{(-k)}$ denotes the estimate with k^{th} replication deleted. The confidence intervals can be calculated with these observations based on Student's t-distribution.

Finally, examine Matlab's functionality for fitting regression metamodels, functions `regress` and `regstats`. Compare the earlier results to the confidence intervals given by these functions.

4.2

Consider a single-server queueing model where customers arrive at exponentially distributed inter-arrival times with rate λ. Service times follow a normal distribution with mean μ and variance σ^2 (we use a truncated distribution where negative values are not allowed). Assume that we can control the rate of the server, i.e., the mean service time μ. Further, let the cost of running 100 customers through the system be

$$J(\mu) = E[D_{100}(\mu, \omega)] + \frac{c}{\mu}$$

where D_{100} is the mean queueing delay of the 100 customers, ω a sample path (I.e. stochastic effects of the system), and *c* a cost factor. Use *c*=2, σ^2=0.01 and $\mu \in$ [0.5, 1] and λ=1.

Your task is to apply a stochastic approximation algorithm to find a value of μ that minimizes *J*. The algorithm is described as follows:

$$\mu^{n+1} = \Pi_M[\mu^n - a_n \hat{\nabla} J(\mu^n)]$$

μ^n	Current solution
Π_M	Projection to fesible set
$\hat{\nabla} J(\mu^n)$	Gradient estimate
a_n	Step size

Compare the following gradient estimation techniques in optimizing the system:

1. Finite difference estimation using independent sampling
2. Finite difference estimation using correlated random numbers
3. Infinitesimal perturbation analysis

Finite difference estimates of the gradient are calculated by estimating the objective function at μ and $\mu+\delta$ through a sufficient number of simulation replications:

$$\hat{\nabla} J(\mu) = \frac{\hat{J}(\mu+\delta) - \hat{J}(\mu)}{\delta}$$

For infinitesimal perturbation analysis (IPA), note that a perturbance of δs in the service time of customer *i-1* will change the queueing delay of customer *i* as follows:

$$\delta d_i = \begin{cases} 0 & \text{if the server is idle} \\ \delta s_{i-1} & \text{if the server is non-idle, but queue is empty} \\ \delta s_{i-1} + \delta d_{i-1} & \text{if the queue is not empty} \end{cases}$$

We record the perturbation in queueing delay for each arriving customer while simulating the system. As service time perturbation, we can simply use the derivative of service time with respect to the change in its mean value. For normally distributed service times, this is $\delta s_i = 1$.

Thus, record for each arriving customer:

$$\delta d_i = \begin{cases} 0 & \text{if the server is idle} \\ 1 & \text{if the server is non-idle, but queue is empty} \\ 1 + \delta d_{i-1} & \text{if the queue is not empty} \end{cases}$$

The gradient estimate based on a single simulation replication then becomes:

$$\hat{\nabla} J(\mu) = \frac{1}{100} \sum_{i=1}^{100} \delta d_i - \frac{c}{\mu^2}$$

We might use this as the gradient in our stochastic approximation algorithm or replicate the simulation a few times and average to obtain an estimate that is based on a larger sample.

In particular, observe how the stochastic approximation algorithm performs with the three gradient estimation techniques, when the number of replications used in the estimation is changed.