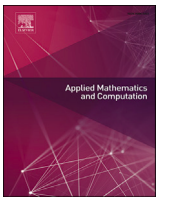
Applied Mathematics and Computation 472 (2024) 128637



Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc



Full Length Article



On the numerical solution of functional equations with application to response time distributions

Peter G. Harrison

Department of Computing, Imperial College London, South Kensington Campus, London SW7 2AZ, UK

A B S T R A C T

A unified approach is developed to solve functional equations defining generating functions. Such equations are often constructed as a means to solve recurrence relations, such as those arising from queue length probabilities and response time probability distributions in Markov models. Many such equations have been obtained over several decades. Some have been solved analytically, some numerically and for some no tractable solution has been found. Our unified approach is able to provide accurate numerical solutions to such equations, even when they include derivatives of the generating function. It solves the JSQ model with two queues for the first time, utilizing a novel "partial" generating function related to the one the functional equation defines. Numerical results, displayed in tables of moments and graphs of probability density functions, show good accuracy against simulations with 500 000 regenerative cycles.

1. Introduction

Generating functions provide a powerful means to transform discrete, infinite recurrence relations in one or more dimensions into a problem in the domain of real or complex analysis. To illustrate, in the one dimensional case, the idea is to encode required numbers a_n, say, as coefficients in a Taylor series $g(x) = \sum_{n=0}^{\infty} a_n x^n$ and then transform the recurrence formula into an equation for $g(x)$. When $g(x)$ is analytic, the coefficients a_n can be recovered by differentiation at the origin or by integrating the product of the generating function and a complex exponential around a contour encircling the origin. However, usually it is not the individual coefficients that are required but weighted sums, for example the mean value or a higher moment of a probability distribution for which $g(x)$ is the probability generating function (pgf).

An important application in performance modelling investigates response time distributions at equilibrium. A recurrence relation is constructed for the Laplace transform of the probability density function of response time, given the state-vector $\vec{s}$ of the system initially, $w^*_{\vec{s}}(\theta)$ say. This may be solved by the generating function method to give the function $g(x_1, \ldots, x_m; \theta)$ where θ is a parameter, considered fixed, and m is the length of $\vec{s}$. In product-form problems, the equilibrium probability that the initial state is $\vec{n}$ is proportional to $\prod_{i=1}^{m} \rho_i^{n_i}$, for certain positive real numbers $\rho_i < 1$, whereupon the required unconditional response time density has Laplace transform proportional to $g(\rho_1, \ldots, \rho_m; \theta)$.

Papers by Verloop, Ayesta and Núñez-Queija (2011); Harrison and Pitel (1995,1993); Fayolle, Flajolet and Hofri (1986); Coffman et al. (1986); Mitra and Morrison (1983); Fayolle, Iasnogorodski and Mitrani (1983), Taube-Netto (1977), amongst many others, all obtain functional equations and each takes a case-specific approach to solving them [2,12,11,5,4,17,6,19]. We introduce a uniform, generic method to solve functional equations for generating functions which are not only of wide scope, but also finds new solutions to problems that were successfully posed as functional equations but were not numerically tractable and so remained unsolved. We find an exact solution (up to numerical computation) for the join-shortest-queue problem (JSQ) with two queues, for the first time (to our knowledge), with obvious generalization to a wider class of similar problems, for example in load balancing [16,7].

E-mail address: pgh@ic.ac.uk.

https://doi.org/10.1016/j.amc.2024.128637
Received 28 October 2023; Received in revised form 19 February 2024; Accepted 25 February 2024

Available online 5 March 2024
0096-3003/© 2024 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY-NC license (http://creativecommons.org/licenses/by-nc/4.0/).

The approach taken is simple in comparison to previous methods used in the papers cited above that use boundary value problems (BVPs) in the complex plane. For functions of one variable, a disk centred on the origin, often the unit disk, is selected as a contour and the unknown generating function, g say, is discretized at n points around its boundary and then represented by a linear interpolation. Values of g at points inside the disk are then given by Cauchy's integral formula (see, for example, [18]). The equation defining the generating function is then applied at each point on the boundary to give a set of n equations in the n unknown values of g at the chosen points. The method extends conveniently to defining equations that also contain derivatives – arising in problems with processor-sharing (PS) queueing discipline, for example. Of course, in one dimension, a function of a single variable can be solved directly and more accurately using the analyticity of the generating function in the disk, as in [6,11,13] for example. However, in higher dimensions, i.e. for functions of more than one variable, this method is, in general, insufficient and following the same approach leads to the BVP-based solutions referred to above. Our method, on the other hand, just extends the disk used for one complex variable to a polydisk (Cartesian product of disks), together with the obvious generalization of Cauchy's integral formula [14]. Although the methodology would, in principle, apply to problems of any dimension, i.e. generating functions of $m \geq 1$ variables, to date, numerical tractability restricts our models to 2 dimensions, $m = 2$ variables.

Before considering the two dimensional problem, we first review and develop the univariate case, partly by way of introduction and partly because in many bivariate cases the internal points referred to above have one variable on a boundary, given by the "chosen points", so that Cauchy's integral formula only needs to be applied to a contour in the complex plane of the other variable.

2. Univariate setup

In the simplest form of the problem, an equation is solved for a function of one variable. This also includes functions of several variables when only one of them changes across the instances of the function appearing in the equation, the others being treated as constant parameters: for example, the equation in the function-variable f,

$$a(x,y)f(x,y) = b(x,y) + c(x,y)f(0,y) \tag{1}$$

where $a(x,y)$, $b(x,y)$ and $c(x,y)$ are known functions of x and y. However, the two-variable equation

$$a(x,y)f(x,y) = b(x,y) + c(x,y)f(0,y) + d(x,y)f(x,0) \tag{2}$$

requires a fuller analysis since neither of the variables is the same in every instance of f in equation (2); this is the main focus of the paper from section 3 onwards.

Univariate problems of the kind in equation (1) have been routinely investigated and are often easily solved by direct methods using analyticity of f – wherever $a(x,y) = 0$, provided f is analytic, the right hand side (RHS) must also vanish, giving an equation for $f(0,y)$. Nevertheless, to illustrate our method, we first describe how it works in this simple case, with less clutter of notation and terminology. Generalization to two or more variables, corresponding to Markov chains with state space dimension higher than one, is straightforward in principle and described in the next main section. However, for reasons of numerical tractability, we restrict ourselves to two dimensions.

2.1. Approximation of the function on a contour

Consider a function of one variable, $G(z)$, defined by a functional equation, which is analytic inside and on some contour γ in the complex plane. For example, if G is a probability generating function, it is analytic inside and on the unit circle. The function G may be discretized on the contour by choosing an ordered set of n points $z_1, \ldots, z_n$ around γ, for some integer n, and defining the variables $g_i = G(z_i)$, for $i = 1, \ldots, n$. In this way G can be approximated on γ by its given values at the discretization points together with interpolation. We say the vector $\vec{g} = (g_1, \ldots, g_n)$ is a *vector representation* of the function G, so that an interpolation $\mathfrak{I}(\vec{g})$ on the pairs $\{(z_i, g_i) | i = 1, \ldots, n, 1\}$ is also represented by $\vec{g}$. $\mathfrak{I}(\vec{g})$ therefore approximates G and, unless otherwise stated, uses *linear* interpolation.

The defining equation of the generating function can now be applied at each of the points z_i and the value of $G(z_0)$ at any internal point z_0 (inside γ) calculated using Cauchy's integral formula (see Theorem 5 in Appendix 6.1), integrating symbolically the function defined by the linear interpolation of $\{g_1, \ldots, g_n\}$ around γ. This gives n equations in the n unknowns g_i, which are solved numerically, subject to any normalization condition, such as $G(1) = 1$ when G is a pgf. In general, some iteration will be necessary, increasing the number of points n to achieve acceptable precision. Typically we found $n = 64$ to be sufficient for several decimal places of accuracy but the choice obviously depends on the specific problem at hand; see Appendix 6.5 for more details.

Having obtained estimates for the values g_i, the function $G(z)$ is approximated by the interpolation $\mathfrak{I}(\vec{g})$ on the contour γ. From the interpolation on γ, the function G and any of its derivatives can be estimated at an interior point by Cauchy's differentiation theorem (Theorem 6 in Appendix 6.1).

2.2. Matrix representation

Working on the functional equation with the vector representation of G leads to complex symbolic expressions in the variables g_i, and so inefficient computations. It is much more efficient and concise to work with matrix representations that lead to entirely numerical calculations in solving for these g_i.

Definition 1. Let $\mathcal{V}$ be the vector space with basis $\{g_1, g_2, \dots, g_n\}$ and the usual arithmetic vector addition and scalar multiplication operators, where the g_i are independent variables ($1 \le i \le n$).

In the standard basis, $g_i = (0, \dots, 0, 1, 0, \dots, 0)^T$ (a column vector), where the 1 occurs in the i^{th} position, and so the vector representation of the function G is the identity matrix $I^{n\times n}$.

Let two functions E, F that are analytic in γ have vector representations $\vec{e}, \vec{f}$, respectively. A linear map on $\mathcal{V}$ given by the matrix A such that $\vec{f} = A\vec{e}$ is a *matrix-transformation* between $\vec{e}$ and $\vec{f}$ that represents the mapping from $E \mapsto F$.

Example: Multiplication by a known function. Suppose F is defined by $F(x) = A(x)E(x)$ for known analytic function A. Then, evaluating $F(x)$ at the discretization points, $\vec{f} = \text{Diag}(\vec{a})\,\vec{e}$, where $\vec{a}$ is the vector representation of the function A. Hence the matrix transformation corresponding to multiplication by a known function $A(x)$ is $\text{Diag}(\vec{a})$, i.e. the diagonal matrix with the vector representation of A on its diagonal.

The benefit of using matrices rather than vector representations is that it is not necessary to manipulate symbolic expressions containing the variables g_i, which become very complex with a large number (n) of variables. The functional equations we consider are linear in G because interpolation and differentiation are linear, as is evaluation of a Cauchy integral to compute a value of $G(z_0)$ at a point z_0 inside γ, for example. All these transformations of a function can therefore be represented by a matrix, T say, that premultiplies the vector representation of the function being transformed – such as $\vec{g}$, represented by $I^{n\times n}$, or a previous transformation thereof – in the vector space $\mathcal{V}$. The benefit comes from the fact that T is a constant matrix.

2.3. Representation of derivatives at internal points by Cauchy integrals

The value of the interpolation of any function F, with vector representation $\vec{f}$, at a point z on γ is a weighted sum of its values f_j at the discretization points z_j ($1 \le j \le n$) – actually at just two adjacent points in a linear interpolation. In the linear case, for example, we could write as an interpolated approximation to F, $\mathfrak{I}(\vec{f})(z) = f_j + \left(\frac{f_{j+1}-f_j}{z_{j+1}-z_j}\right)(z - z_j)$ for all z considered to lie between the points z_j and z_{j+1}, where we define $z_{n+1} = z_1; f_{n+1} = f_1$. However, with a circular integration path, it is more precise to write

$$\mathfrak{I}(\vec{f})(re^{i\theta}) = f_j + \left(\frac{f_{j+1} - f_j}{\theta_{j+1} - \theta_j}\right)(\theta - \theta_j)$$

for $\theta_j \le \theta \le \theta_{j+1}$, where $\theta_j = \text{Arg}\, z_j$. In this way the interpolation is over a real domain, which is more efficient and widely supported in mathematical software. Obviously we would get a more complex expression at higher orders of interpolation, but it would still be linear in the f_j; we stick to linear interpolation in symbolic integrals.

We can now estimate derivatives at internal points by applying Cauchy's differentiation theorem (Theorem 6 in Appendix 6.1):

Proposition 1. *Let the contour γ be the circle centred at the origin with radius r. The value $F^{(d)}(w)$ of the d^{th} derivative of an analytic function F at a point w inside γ is approximated by $\vec{k}_w^{(d)}.\vec{f}$, where $\vec{f}$ is a vector representation of F on γ at discretization points $\{z_1, \dots, z_n\} \subset \gamma$ and the j^{th} component ($1 \le j \le n$) of the vector $\vec{k}_w^{(d)}$ is:*

$$k_{w;j}^{(d)} = \frac{d!}{2\pi r^d}\left[\left(\frac{1}{\theta_{j+1} - \theta_j}\right)\int_{\theta_j}^{\theta_{j+1}} \frac{\theta_{j+1} - \theta}{(e^{i\theta} - w/r)^{d+1}} e^{i\theta} d\theta + \left(\frac{1}{\theta_j - \theta_{j-1}}\right)\int_{\theta_{j-1}}^{\theta_j} \frac{\theta - \theta_{j-1}}{(e^{i\theta} - w/r)^{d+1}} e^{i\theta} d\theta\right]$$

with error $\Theta(n^{-3})$, where $f_j = F(re^{i\theta_j}), \theta_j = \text{Arg}\, z_j$ ($1 \le j \le n$), $\theta_0 = \theta_n - 2\pi$ and $\theta_{n+1} = 2\pi + \theta_1$.

Proof. Under linear interpolation on the argument of z, the d^{th} derivative at w is approximated by

$$\begin{aligned}
\frac{d!}{2\pi i}\oint_\gamma \frac{\mathfrak{I}(\vec{f})(z)}{(z-w)^{d+1}} dz &= \frac{d!}{2\pi i}\sum_{j=1}^{n} f_j \int_{\theta_j}^{\theta_{j+1}} \frac{rie^{i\theta} d\theta}{(re^{i\theta} - w)^{d+1}} + \left(\frac{f_{j+1} - f_j}{\theta_{j+1} - \theta_j}\right)\int_{\theta_j}^{\theta_{j+1}} \frac{\theta - \theta_j}{(re^{i\theta} - w)^{d+1}} rie^{i\theta} d\theta \\
&= \frac{d!}{2\pi i}\sum_{j=1}^{n} f_j \left[\int_{\theta_j}^{\theta_{j+1}} \frac{rie^{i\theta} d\theta}{(re^{i\theta} - w)^{d+1}} - \left(\frac{1}{\theta_{j+1} - \theta_j}\right)\int_{\theta_j}^{\theta_{j+1}} \frac{\theta - \theta_j}{(re^{i\theta} - w)^{d+1}} rie^{i\theta} d\theta\right] \\
&\quad + \sum_{j=1}^{n}\left(\frac{f_{j+1}}{\theta_{j+1} - \theta_j}\right)\int_{\theta_j}^{\theta_{j+1}} \frac{\theta - \theta_j}{(re^{i\theta} - w)^{d+1}} rie^{i\theta} d\theta \\
&= \frac{d!}{2\pi i}\sum_{j=1}^{n}\left(\frac{f_j}{\theta_{j+1} - \theta_j}\right)\int_{\theta_j}^{\theta_{j+1}} \frac{\theta_{j+1} - \theta}{(re^{i\theta} - w)^{d+1}} rie^{i\theta} d\theta
\end{aligned}$$

$$+\frac{d!}{2\pi i}\sum_{j=2}^{n+1}\left(\frac{f_j}{\theta_j-\theta_{j-1}}\right)\int_{\theta_{j-1}}^{\theta_j}\frac{\theta-\theta_{j-1}}{(re^{i\theta}-w)^{d+1}}rie^{i\theta}d\theta$$

and the expression for $k^{(d)}_{w;j}$ follows, given that $\theta_0=\theta_n-2\pi$ and $\theta_{n+1}=2\pi+\theta_1$.

The error terms in the linear Taylor-approximations of $F(re^{i\theta})$ are (by the Mean Value Theorem)

$$\frac{F^{(2)}(re^{i\theta^*})}{2}\int_{\theta_j}^{\theta_{j+1}}\frac{(\theta-\theta_j)^2}{(re^{i\theta}-w)^{d+1}}rie^{i\theta}d\theta=\frac{F^{(2)}(re^{i\theta^*})ri}{2e^{id\theta_j}}\int_0^{\theta_{j+1}-\theta_j}\frac{\theta^2e^{i\theta}}{(re^{i\theta}-we^{-i\theta_j})^{d+1}}d\theta$$

for some $\theta^*\in[\theta_j,\theta_{j+1}]$.

The integrand has order $\Theta(\theta^2)$ and $F^{(2)}(re^{i\theta^*})$ is bounded by $\sup_{\theta_j\le\theta\le\theta_{j+1}}\frac{d^2F(re^{i\theta})}{d\theta^2}<\infty$. Hence the integrals are all $\Theta(n^{-3})$ because $\theta_{j+1}-\theta_j$ is $\Theta(n^{-1})$. □

When solving generating function equations, we need to represent the value at an internal point for every discretization point (the same value for each). To this end we define the matrix $K^{(d)}_w$ to represent the linearized mapping between $F(z)$ and the constant function $F^{(d)}(w)$:

Definition 2. The $n\times n$ matrix $K^{(d)}_w$ comprises $\vec{k}^{(d)}_w$ for each of its row-vectors.

2.4. Representation of derivatives on the contour

Some functional equations include derivatives that also need to be estimated on the contour as symbolic expressions to get a solution for the variables g_i; an instance of this is [4], considered in section 4.2.

The direct method of central differencing estimates the first derivative with respect to z on the contour γ by the fraction $\frac{g_{i+1}-g_{i-1}}{z_{i+1}-z_{i-1}}$ at each point $z_i\in\gamma$.

Definition 3. The *central differencing derivative matrix-transformation* Δ is defined to be the matrix obtained by subtracting the identity matrix rotated left from the identity matrix rotated right, then dividing each row j elementwise by the corresponding increment in the variable z, i.e. $z_{j+1}-z_{j-1}$, for $j=1,\ldots,n$, where $z_0=z_n, z_{n+1}=z_1$. In pseudo-code,

$$\Delta=\frac{\text{RotateRight}/@\,\boldsymbol{I}^{n\times n})-\text{RotateLeft}/@\,\boldsymbol{I}^{n\times n}}{\text{RotateLeft}[\vec{z}]-\text{RotateRight}[\vec{z}]}$$

where $/@$ denotes the map-function and the division operation is of the rows of the matrix in the numerator by the corresponding components of the vector in the denominator.

The first derivative of $G(z)$ is estimated by the vector $\Delta\vec{g}$ at the discretization points, which is therefore a vector representation of $G'(z)$ as required.

Unfortunately, this results in the set of n derivative estimates being linearly dependent, since the points z_j are cyclic around γ. This is easily seen by noting that $\sum_{i=1}^n(z_{i+1}-z_{i-1})(\Delta\vec{g})_i=0$. This may lead to a set of linear equations for the generating function's values at the discretization points that are not full rank, and so have no unique solution.

Instead, our approach is to use Cauchy's differentiation theorem on a distorted contour C, obtained by replacing a small arc of γ by the arc of a circle (approximately a semi-circle) centred at the middle of the removed arc and meeting the remainder of γ at its endpoints. This is illustrated in Fig. 1 for the common case that γ is a circle centred on the origin and the detailed calculation is laid out in Theorem 1 below.

Working symbolically with a given set of points on the contour γ, $\{z_i\mid 1\le i\le n\}$, we obtain an approximation for the dth derivative $G^{(d)}(z)$ at every point z_i for $d\ge 0$, solely in terms of the variables $\{g_i\mid 1\le i\le n\}$, which we write as

$$\vec{g}^{(d)}=(\ldots,g_i^{(d)},\ldots)=M^{(d)}(\vec{g})\quad(1\le i\le n)\tag{3}$$

for some vector-valued linear function $M^{(d)}$.

In fact there is a more important application for equation (3) when $d=0$ that we will use in section 3.2, where we deal with "partial" generating functions in two dimensions that sum over the semi state spaces $\{(i,j)\mid 0\le i\le j\}$ or $\{(i,j)\mid 0\le j\le i\}$.

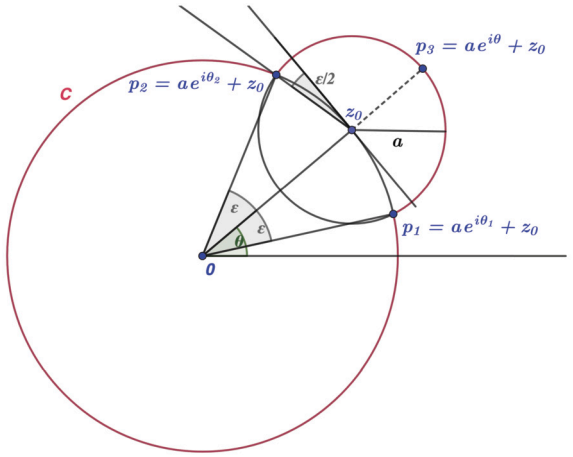


Fig. 1. The distorted contour of integration C.

2.4.1. Determination of the mapping functions $M^{(d)}$

Consider $z_0 \in \partial D$, the boundary of the disk $D \equiv D(0,r)$, centred at the origin with radius r.[1] Applying Cauchy's differentiation theorem (Theorem 6 in Appendix 6.1):

$$G^{(d)}(z_0) = \frac{d!}{2\pi i} \oint_C \frac{G(z)}{(z-z_0)^{d+1}} dz, \tag{4}$$

where C is the distorted boundary of ∂D shown in red in Fig. 1: the part of ∂D that goes anticlockwise from p_2 to p_1, then around the arc of the small circle centred at z_0 with small radius a from p_1 to p_2 via p_3. Let $\text{Arg}(z_0) = \theta$ so that $z_0 = re^{i\theta}$, $p_1 = z_0 + ae^{i\theta_1}$, $p_2 = z_0 + ae^{i\theta_2}$ and $p_3 = z_0 + ae^{i\theta}$, where $\theta_1 = \theta - (\pi+\epsilon)/2$, $\theta_2 = \theta + (\pi+\epsilon)/2$ and $a = 2r\sin\frac{\epsilon}{2}$ by basic geometry. We now have the following result for computing the derivatives of G at z_0.

Theorem 1. *For $\epsilon > 0$ such that a function G is analytic inside and on the distorted contour C, the derivatives of G at a point $z_0 \in \partial D$ are given by:*

$$\frac{(\pi-\epsilon)i}{d!} G^{(d)}(z_0) = I_d + \sum_{\substack{k=-d \\ k\neq 0}}^{\infty} \frac{G^{(k+d)}(z_0)}{(k+d)!} \frac{(2iz_0(\sin\frac{\epsilon}{2}))^k}{k} \begin{cases} 2i\sin\frac{k\epsilon}{2} & \text{if } k \text{ even} \\ 2\cos\frac{k\epsilon}{2} & \text{if } k \text{ odd} \end{cases}$$

where $I_d = \int_{p_2}^{p_1} \frac{G(z)}{(z-z_0)^{d+1}} dz$ for $d \geq 0$.

Proof. The integral along the arc from p_1 to p_2 of the small circle is, using the Taylor series expansion for $G(z)$ at $z = z_0$,

$$\int_{\theta-\frac{\pi+\epsilon}{2}}^{\theta+\frac{\pi+\epsilon}{2}} \frac{G(z_0+ae^{i\phi})}{a^{d+1}e^{i(d+1)\phi}} iae^{i\phi} d\phi = \int_{\theta-\frac{\pi+\epsilon}{2}}^{\theta+\frac{\pi+\epsilon}{2}} \frac{\sum_{k=0}^{\infty} G^{(k)}(z_0)a^k e^{ik\phi}/k!}{a^{d+1}e^{i(d+1)\phi}} iae^{i\phi} d\phi$$

$$= i\sum_{k=0}^{\infty} G^{(k)}(z_0) A(k-d)/k!$$

where $A(m) = a^m \int_{\theta-\frac{\pi+\epsilon}{2}}^{\theta+\frac{\pi+\epsilon}{2}} e^{im\phi} d\phi$. Thus $A(0) = \pi + \epsilon$ and for $m \neq 0$,

$$A(m) = \frac{a^m e^{im\theta}}{im}\left[e^{im(\pi+\epsilon)/2} - e^{-im(\pi+\epsilon)/2}\right] = \frac{a^m e^{im\theta} i^m}{im}\left[e^{im\epsilon/2} - (-1)^m e^{-im\epsilon/2}\right]$$

$$= \frac{2^m i^{m-1} z_0^m (\sin\frac{\epsilon}{2})^m}{m} \begin{cases} 2i\sin\frac{m\epsilon}{2} & \text{if } m \text{ even} \\ 2\cos\frac{m\epsilon}{2} & \text{if } m \text{ odd} \end{cases}$$

Hence, from equation (4),

[1] $D(c,r)$ denotes a disk of radius r centred at c, $(r \in \mathbb{R}, c \in \mathbb{C})$. For example, $D(0,1)$ is the unit disk centred at the origin.

$$2\pi i G^{(d)}(z_0) = d! \int_{p_2}^{p_1} \frac{G(z)}{(z-z_0)^{d+1}} dz + G^{(d)}(z_0)(\pi+\epsilon)i$$
$$+ d! \sum_{\substack{k=-d \\ k\neq 0}}^{\infty} \frac{G^{(k+d)}(z_0)}{(k+d)!} \frac{(2iz_0(\sin\frac{\epsilon}{2}))^k}{k} \begin{cases} 2i\sin\frac{k\epsilon}{2} & \text{if } k \text{ even} \\ 2\cos\frac{k\epsilon}{2} & \text{if } k \text{ odd} \end{cases}$$

and the result follows. □

Remarks.

1. The integral from p_2 to p_1 has no singularities on its path (since the generating function G is analytic on C) and is approximated symbolically in terms of $\{g_1, \ldots, g_n\}$, using linear interpolation, as in Section 2.3.
2. Since integration is a linear operator, the expressions computed by the Cauchy differentiation formula $\int_{p_2}^{p_1} \frac{I(\vec{g})(z)}{(z-z_0)^{d+1}} dz$ are also linear in the g_i and so the functional $M^{(d)}$ can be represented by a *matrix* M_d to premultiply $\vec{g}$. The interpolation for $G^{(d)}$ on ∂D is then given by the vector $M_d\vec{g}$ corresponding to the points $\vec{g}$, for $d \geq 0$.
3. In the case that $d = 0$, we find $\vec{g} = M_0\vec{g}$, so that g is a fixed point of M_0. Normally, however, we use the identity matrix to represent the transformation from G to itself, i.e. assume $M_0 = I^{n\times n}$.
4. Notice that equation (3) is satisfied *for all* analytic functions G defined inside D. Therefore, given the discretization points, the linear maps $M^{(d)}$ (and so matrices M_d) can be applied to the vector representation of any analytic function of z on ∂D. They may also be reused for different problems, which is a major advantage since for a fine mesh size, the calculation of M_d is expensive, especially when extended to two or more dimensions.
5. The first order term can be omitted if ϵ can be chosen to be arbitrarily small, i.e. set to 0. However, for derivatives, ϵ^{-d} becomes very large for small ϵ, possibly leading to overflow. A somewhat larger value for ϵ, up to the size of a discretization interval's argument increment, $2\pi/n$, suffices if the first order terms are included, leading to an obvious trade-off; we found that a fraction of the discretization interval's angle in the region $[\frac{1}{3}, \frac{3}{4}]$ gave good results when a derivative was involved in the functional equation; see Section 6.5 for more details.
6. Higher derivatives in the functional equation quickly run into numerical problems because of the potentially very large terms with factors ϵ^{-d} – of order at least $\Theta(n^d)$ since ϵ must be less than an interval-angle. Correspondingly, the numerical integrands diverge to order ϵ^{-d-1} near the limits of the integration, p_1 and p_2, and rely on cancellation with the other ϵ^{-d} terms (see the proof of Proposition 2). This leads to numerical tractability issues when d is 2 or more. However, for first derivatives, these terms are only of order $\Theta(n)$, typically about 100, and cancel well, giving good results in the examples we have tried with ϵ chosen between 0.3 and 0.8 discretization interval-angles.

We use the following approximation to second order in ϵ:

Corollary 1. *To order ϵ^3,*

$$G^{(1)}(z_0) = -\frac{i}{\pi-\epsilon}\left[I_1 + \frac{2i(1-\epsilon^2/12)}{z_0\epsilon} G^{(0)}(z_0) + iz_0\epsilon G^{(2)}(z_0)\right]$$

Proof. Expanding the series in Theorem 1 to order ϵ^2 (noting that it contains only odd powers) we have:

$$(\pi-\epsilon)iG^{(1)}(z_0) = I_1 - G^{(0)}(z_0)(2iz_0\sin(\epsilon/2))^{-1}\, 2\cos(\epsilon/2) + G^{(2)}(z_0)iz_0\sin(\epsilon/2)\, 2\cos(\epsilon/2)$$
$$= I_1 + \frac{iG^{(0)}(z_0)\cot(\epsilon/2)}{z_0} + iz_0G^{(2)}(z_0)\sin\epsilon$$

and the result follows on expanding $\cot(\epsilon/2) = 1/\tan(\epsilon/2)$ and $\sin\epsilon$. □

Notice that this corollary alone cannot provide numerical results for $G^{(1)}(z_0)$ since it involves the unknown quantities $G^{(0)}(z_0), G^{(2)}(z_0)$. However, we can estimate these by the matrices $I^{n\times n}, \Delta^2$ respectively.

We are now in a position to determine a matrix to represent the transformation corresponding to differentiation on the contour C.

Proposition 2. *A linear interpolated vector representation of the first derivative of a function $F(z)$ that is analytic on and inside the contour C is $M_1\vec{f}$, where $\vec{f}$ is a vector representation of F on C at discretization points $\{z_1, \ldots, z_n\} \subset C$, and the matrix M_1 is, for a certain matrix $\tilde{K}^{(1)}$ defined in the proof:*

$$M_1 = -\frac{i}{\pi-\epsilon}\left(\tilde{K}^{(1)} + \frac{2i(1-\epsilon^2/12)}{z_0\epsilon} I^{n\times n} + iz_0\epsilon\Delta^2\right)$$

with error $\Theta(n^{-1})$.

Proof. Let the discretization points be $\{z_1,\dots,z_n\}\subset C$ and let $\theta_j=\arg z_j$ for $1\le j\le n$. In addition, define $\phi_1=\arg=\theta_1+\epsilon$, $\phi_{n+1}=\theta_1+2\pi-\epsilon$ and $\phi_j=\theta_j$ for $2\le j\le n$, where $0<\epsilon<2\pi/n$ — see Fig. 1 with $z_0=z_1$. Then the linear interpolated approximation to $F(z)$ for $z\in C$ is

$$\Im(\vec{f})(re^{i\phi})=f_j+\left(\frac{f_{j+1}-f_j}{\theta_{j+1}-\theta_j}\right)(\phi-\theta_j)\quad\text{for }\phi_j\le\phi\le\phi_{j+1},1\le j\le n\ \text{(recalling that } f_{n+1}=f_1).$$

The integral corresponding to I_d of Corollary 1 has linearized value (as in the proof of Proposition 1):

$$\begin{aligned}I_d&=\int_{\phi_1}^{\phi_{n+1}}\frac{\Im(\vec{f})(re^{i\phi})rie^{i\phi}}{(re^{i\phi}-re^{i\theta_1})^{d+1}}d\phi\\&=\sum_{j=1}^{n}\left(\frac{f_j}{\theta_{j+1}-\theta_j}\right)\int_{\phi_j}^{\phi_{j+1}}\frac{\theta_{j+1}-\phi}{(re^{i\phi}-re^{i\theta_0})^{d+1}}rie^{i\phi}d\phi\\&\quad+\sum_{j=2}^{n+1}\left(\frac{f_j}{\theta_j-\theta_{j-1}}\right)\int_{\phi_{j-1}}^{\phi_j}\frac{\phi-\theta_{j-1}}{(re^{i\phi}-re^{i\theta_0})^{d+1}}rie^{i\phi}d\phi\\&=\vec{\kappa}_1^{(d)}(\theta).\vec{f}\end{aligned}$$

where the vector $\vec{\kappa}_1^{(d)}(\theta)$ is defined by its j^{th} component ($1\le j\le n$):

$$\kappa_{1;j}^{(d)}=\left(\frac{i}{r^d}\right)\times\begin{cases}\left(\frac{1}{\theta_{j+1}-\theta_j}\right)\int_{\theta_j}^{\theta_{j+1}}\frac{\theta_{j+1}-\theta}{(e^{i\theta}-e^{i\theta_0})^{d+1}}e^{i\theta}d\theta+\left(\frac{1}{\theta_j-\theta_{j-1}}\right)\int_{\theta_{j-1}}^{\theta_j}\frac{\theta-\theta_{j-1}}{(e^{i\theta}-e^{i\theta_0})^{d+1}}e^{i\theta}d\theta & (2\le j\le n)\\ \left(\frac{1}{\theta_2-\theta_1}\right)\int_{\theta_1}^{\theta_2}\frac{\theta_2-\theta-\epsilon}{(e^{i\theta}-e^{i\theta_0})^{d+1}}e^{i\theta}d\theta+\left(\frac{1}{\theta_{n+1}-\theta_n}\right)\int_{\theta_n}^{\theta_{n+1}}\frac{\theta-\theta_n}{(e^{i\theta}-e^{i\theta_0})^{d+1}}e^{i\theta}d\theta & (j=1)\end{cases}\tag{5}$$

Notice that for the component $j=1$, we have the sum of two contributions: the coefficient of f_1 and the coefficient of $f_{n+1}\equiv f_1$.

The cyclic numbering of the discretization points z_i is arbitrary, i.e. we can choose any of them to be z_1. Therefore, to compute the value of I_d at the point $z_0=z_2$ it is only necessary to recalculate $\vec{\kappa}_1^{(d)}$ with the discretization points rotated to the left – we call this vector $\vec{\kappa}_2^{(d)}$. By rotating left successively, we obtain similarly $\vec{\kappa}_3^{(d)},\dots,\vec{\kappa}_n^{(d)}$ and denote by $\tilde{K}^{(d)}$ the matrix with i^{th} row equal to $\vec{\kappa}_i^{(d)}$ ($1\le i\le n$).

Here we choose $d=1$ and in the other terms of Corollary 1, we use $\vec{f}$ and the approximation $\Delta^2\vec{f}$ to represent the terms $F^{(0)}(z_0)$ and $F^{(2)}(z_0)$ respectively.[2] We therefore approximate $F^{(1)}(z_0)=M_1\vec{f}$ for discretization point $z_0\in C$, where

$$M_1=-\frac{i}{\pi-\epsilon}\left(\tilde{K}^{(1)}+\frac{2i(1-\epsilon^2/12)}{z_0\epsilon}I^{n\times n}+iz_0\epsilon\Delta^2\right).$$

This completes the proof apart from the error term.

Following the same reasoning as in the proof of Proposition 1, with w replaced by $re^{i\theta_0}$, the errors in the linear Taylor-approximations of $F(re^{i\theta})$ are $\Theta(n^{-3})$ in all the integrals that do not have θ_1 or θ_{n+1} for either limit.

The errors in the integrals with θ_1 or θ_{n+1} for one of their limits, corresponding to the components $\kappa_{1;1}^{(d)},\kappa_{1;2}^{(d)}$ and $\kappa_{1;n}^{(d)}$ in equation (5), are estimated as follows. Without loss of generality, we assume that $\theta_0=0$, so that $\theta_1=\epsilon$. Then the error term corresponding to $\kappa_{1;1}^{(1)}$ is, cf. Proposition 1,

$$\frac{F^{(2)}(re^{i\theta^*})i}{2r^d}\int_{\epsilon}^{\theta_2}\frac{(\theta-\epsilon)^2e^{i\theta}}{(e^{i\theta}-1)^{d+1}}d\theta=\frac{F^{(2)}(re^{i\theta^*})i}{2r^d}\int_{\epsilon}^{\theta_2}\frac{(\theta-\epsilon)^2e^{-i(d-1)\theta/2}}{(2i\sin(\theta/2))^{d+1}}d\theta$$

for some $\theta^*\in[0,\theta_2]$. This integral can be evaluated symbolically for $d=1$, working to third order in θ, and shown to be $\Theta(\theta_2)=\Theta(n^{-1})$). The component $\kappa_{1;n}^{(1)}$ is similar.

Finally, for the small sector of the contour around z_0, the error is $\Theta(\epsilon^3)$ by Corollary 1, which is $O(n^{-3})$ since we must have $\epsilon<2\pi/n$. Hence the error in the matrix M_1 is $\Theta(n^{-1})$). □

[2] There are better approximations for the second derivative, but we found these gave little improvement in our numerical calculations.

Table 1
Matrices representing operations in the summands of the functional equation for G: G itself; multiplication by a known fixed function $a(z)$; differentiation; and evaluation at a fixed internal point w.

Operation	Matrix Representation	Relevant result
Identity operator giving G	$I^{n\times n}$	Definition 1
Multiplication by known function $A(z)$	Diag$(\vec{a})$	Definition 1 (Example)
Differentiation d times wrt z	M_d	Proposition 2
Value at internal point w	$K_w^{(0)}$	Proposition 1, Definition 2

It is interesting to note that the zero-order contribution in the interpolation,

$$\frac{F^{(0)}(re^{i\theta^*})i}{r}\int_{\epsilon}^{\theta_2}\frac{1}{(2i\sin(\theta/2))^2}d\theta=\frac{F^{(0)}(re^{i\theta^*})i}{2r}\left[\cot(\theta/2)\right]_{\epsilon}^{\theta_2}$$

with a similar contribution from the corresponding integral with upper limit θ_{n+1}. Hence we see that the term in Corollary 1 of order $\Theta(1/\epsilon)$ does indeed cancel out.

2.5. Matrix formulation of the numerical solution

In one dimension, functional equations of the kind defined above can be expressed in terms of the matrix-transformations of the operators tabulated in Table 1 for multiplication, differentiation and evaluation at an internal point of the contour ∂D. The resulting linear equations can then be solved to estimate the values of the function sought at the discretization points, and the function itself approximated by the corresponding interpolation to order 1, 2 or 3. For example, the equation

$$a(x)G(x)=b(x)\frac{\partial}{\partial x}G(x)+c(x)G(0)+d(x)$$

where $a(x)$, $b(x)$ and $c(x)$ are known functions of x, can be represented by the matrix–vector equation:

$$\left(Diag(\vec{a})-Diag(\vec{b}).M_1-Diag(\vec{c}).K_0^{(0)}\right).\vec{g}=\vec{d}$$

where $\vec{a},\vec{b},\vec{c},\vec{d}$ are vector representations of the functions a,b,c,d corresponding to the discretization points used for $\vec{g}$.

3. Multivariate setup

In a two (or higher) dimensional problem, i.e. with two or more variables, there are other operations that need to be represented by matrices, and those defined above need to be extended to include the additional dimensions.

Theorem 7 in Appendix 6.2 gives the analogous formula to Cauchy's for multivariate functions, so we can easily generalize Theorem 1. The joint domain of the variables is a *polydisk*, defined as the Cartesian product $D_x\times D_y$ of individual variables' disks in their own complex planes. In the several variable case the integral is taken over the distinguished boundary of this polydisk, i.e. over a torus (see Definition 6).

3.1. Points in the polydisk or on its boundary

Generalizing the univariate setup to two variables, we can simply apply Theorem 7 in place of Cauchy's integral formula to compute the values of G, and any of its partial derivatives, at an interior point (x_0,y_0). Similarly, when (x_0,y_0) is on the distinguished boundary, we extend the mapping $M^{(0)}$ to $M^{(0,0)}$, with matrix representation $M_{0,0}$, defined in terms of a two-dimensional integral over the distinguished boundary, distorted to include arcs around the points x_0 and y_0 on the constituent disks' boundaries in the x- and y-complex planes. The mapping $M^{(a,b)}$ (and matrix representation $M_{a,b}$) corresponding to the partial derivative $\frac{\partial^{a+b}G}{\partial x^a\partial y^b}$ may be defined similarly by the obvious extension to Theorem 7 if required:

$$\frac{\partial^{a+b}G(z_1,z_2)}{\partial x^a\partial y^b}=(2\pi i)^{-2}a!b!\int_{\partial(D_x\times D_y)}G(\zeta_1,\zeta_2)(\zeta_1-z_1)^{a+1}(\zeta_2-z_2)^{b+1}d\zeta_1d\zeta_2, \tag{6}$$

analogous to Cauchy's differentiation theorem. However, here we will only need the case $a=b=0$.

In the numerical setup, discretization point-pairs $\{(x_i,y_j)|1\le i\le m,1\le j\le n\}$ are now chosen on the distinguished boundary and listed in alphabetic order, producing a vector of mn components. If desired, the matrices $M_{a,b}$ can then be constructed, compatible with this ordering, from their defining double integrals.

More precisely, for any number of variables, we consider the vector space given by the tensor (or, direct) product, $\otimes$, of the discretization point vectors in each dimension, together with the corresponding tensor product defined on the associated linear maps. In the case of two variables, the bilinear map defined by the tensor product is represented by the Kronecker product of the two matrices representing the constituent linear maps, also denoted $\otimes$.

Table 2
Matrices representing operations in the summands of the functional equation for $G(x,y)$: G itself; multiplication by a known fixed function $a(x,y)$; differentiation wrt x; differentiation wrt y; evaluation at a fixed internal point u_0 in the x-disk; evaluation at a fixed internal point v_0 in the y-disk; mappings between discretization sets; lower and upper partial generating functions.

Operation	Matrix	Relevant result
Identity operator giving G	$I^{nm\times nm}$	Definition 1
Multiplication by known function $A(x,y)$	$\Lambda(\vec{a})$	Section 3.1
Differentiation wrt x	$M_{1,0}$	Section 3.1
Differentiation wrt y	$M_{0,1}$	Section 3.1
Values on ∂D_y at a point $u_0 \in D_x$	U_{u_0}	Section 3.1
Values on ∂D_x at a point $v_0 \in D_y$	V_{v_0}	Section 3.1
Mapping $(x,y) \mapsto (x, \lvert x\rvert\, y/x)$	$\Omega^{\leq}$	Section 3.3.1
Mapping $(x,y) \mapsto (\lvert y\rvert\, x/y, y)$	$\Omega^{\geq}$	Section 3.3.1
Partial generating function $G^{\leq}$	$P^{\leq}$	Theorem 2, Proposition 4
Partial generating function $G^{\geq}$	$P^{\geq}$	Theorem 2, Proposition 4

In the problems of practical interest that we consider, however, the generalized operations required are:

- partial differentiation with respect to one variable;
- evaluation at a point that is internal to one variable's disk but on the boundary of the others;
- multiplication by a fixed function of all the variables

The first two of these only require integration around one variable's disk boundary and the third does not involve integration at all. Consequently it is not necessary to compute double (or higher) integrals since we can work in each constituent subspace separately. Focusing on the bivariate problem, we can simply represent the linear operators $M^{(1,0)}$ and $M^{(0,1)}$ by the matrices

$$M_{1,0} = M_1 \otimes I^{n\times n} \quad \text{and} \quad M_{0,1} = I^{m\times m} \otimes M_1$$

respectively.[3]

Similarly, evaluation of $G(x_0, \vec{y})$, when x_0 lies inside the x-disk and $\vec{y}$ is the vector of discretization points on the boundary of the y-disk, has matrix-transformation

$$U_{x_0} = K^{(0)}_{x_0} \otimes I^{n\times n}$$

and when $\vec{x}$ is the vector of discretization points on the boundary of the x-disk and y_0 lies inside the y-disk, $G(\vec{x}, y_0)$ has matrix-transformation

$$V_{y_0} = I^{m\times m} \otimes K^{(0)}_{y_0}.$$

Multiplication by a fixed function $A(x,y)$ is again represented by a diagonal matrix of the function's values at the discretization point-pairs, ordered alphabetically, and we define

$$\Lambda(A) = \mathrm{Diag}\Big(A(x_i, y_j) \mid 1 \leq i \leq m, 1 \leq j \leq n\Big)$$

In this way we obtain the first six entries in Table 2.

3.2. Partial generating functions

Partial generating functions sum over a subset of the reachable states of a two-dimensional Markov process: here, $\{(i,j) \mid 0 \leq i \leq j\}$ or $\{(i,j) \mid 0 \leq j \leq i\}$. They can be determined as Cauchy integrals using simple linear maps.

Definition 4. Given generating function $G(x,y) = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \pi_{i,j} x^i y^j$, the partial generating functions over the lower and upper state spaces are defined, respectively, by

$$G^{\leq}(x,y) = \sum_{0\leq i\leq j} \pi_{i,j} x^i y^j = \sum_{i=0}^{\infty}\sum_{j=i}^{\infty} \pi_{i,j} x^i y^j$$

[3] The dimension of each instance of the matrix M_1 is given by the context: on the left of the Kronecker product it is m whereas on the right it is n.

and

$$G^{\geq}(x,y) = \sum_{0\leq j\leq i} \pi_{i,j} x^i y^j = \sum_{i=0}^{\infty} \sum_{j=0}^{i} \pi_{i,j} x^i y^j.$$

The following result facilitates the solution of a number of unsolved problems like two parallel queues with join-shortest-queue scheduling (see section 4.3) and queues with load balancing.

Lemma 1. *Suppose the function $f(z)$ has a Laurent series $\sum_{i=-\infty}^{\infty} a_i z^i$ in an annulus A centred on the origin, and $C \subset A$ is a contour enclosing the origin. Then for any z_0 in the interior of C:*

$$\frac{1}{2\pi i} \oint_C \frac{f(z)}{z - z_0} dz = \sum_{i=0}^{\infty} a_i z_0^i \tag{7}$$

Proof. For integer $n > 0$, let $I_n = \oint_C \frac{z^{-n}}{z-z_0} dz$. Then

$$\begin{aligned} I_n &= \oint_C z^{-n-1} \left(1 + \frac{z_0}{z - z_0}\right) dz \\ &= z_0 I_{n+1} + \oint_C z^{-n-1} dz = z_0 I_{n+1} \end{aligned}$$

Therefore $z_0 I_{n+1} = I_n = \ldots = z_0^{-n+1} I_1$. Hence

$$\begin{aligned} I_n &= z_0^{-n} \oint_C \left(\frac{1}{z - z_0} - \frac{1}{z}\right) dz \\ &= 0 \end{aligned}$$

since both z_0 and 0 are inside C.

Hence the principal part of the Laurent series of f (the series of terms with negative degree) gives zero contribution and the result follows by Cauchy's integral formula (Theorem 5, Appendix 6.1) applied to the rest of the Laurent series. □

Lemma 2. *If G is a two-parameter generating function that is analytic in the polydisk $D_x \times D_y$ and on its distinguished boundary $\partial D_x \times \partial D_y$, then for any x_0, y_0 with $|x_0| \leq r_1$, $|y_0| \leq r_2$:*

$$G^{\leq}(r_2 x_0/y_0, y_0) = \frac{1}{2\pi i} \oint_{C_y} \frac{G(r_2 x_0/y, y)}{y - y_0} dy \tag{8}$$

and

$$G^{\geq}(x_0, r_1 y_0/x_0) = \frac{1}{2\pi i} \oint_{C_x} \frac{G(x, r_1 y_0/x)}{x - x_0} dx \tag{9}$$

where the contours C_x, C_y are, respectively, the circles $\partial D_x, \partial D_y$ (with respective radii r_1, r_2), or distortions thereof that go round the points x_0, y_0 when $x_0 \in \partial D_x$ or $y_0 \in \partial D_y$; see Fig. 1.

Proof. We just prove the first equation, the second one following by symmetry. We first note that the function $G(r_2 x_0/y, y)$ has a Laurent series on C_y (which has radius r_2) because $r_2|x_0|/|y| = |x_0| \leq r_1$, since $y \in C_y$ and by hypothesis. Thus we have:

$$\begin{aligned} G(r_2 x_0/y, y) &= \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} g_{i,j} (r_2 x_0/y)^i y^j \\ &= \sum_{i=0}^{\infty} \sum_{j=0}^{i-1} g_{i,j} (r_2 x_0)^i y^{j-i} + \sum_{i=0}^{\infty} \sum_{j=i}^{\infty} g_{i,j} (r_2 x_0/y)^i y^j \end{aligned}$$

Since all of its terms have a negative power of y, the first summation on the RHS integrates to zero on the contour ∂D_y by Lemma 1. The second summation is a Taylor series (and hence analytic), so integrates to $G^{\leq}(r_2 x_0/y_0, y_0)$ and the result follows. □

Remarks:

1. The contours C_x and C_y need not be restricted to ∂D_x and ∂D_y. They must lie in the annuli with radii $\{r_1|y_0|/r_2, r_1\}$ and $\{r_2|x_0|/r_1, r_2\}$ respectively, so that the integrands are defined.[4] Thus if x_0 and y_0 lie on the boundaries ∂D_x and ∂D_y, the annuli reduce to these circles.
2. Replacing x_0 by x_0y_0/r_2 in equation (8), we find:

$$G^{\leq}(x_0, y_0) = \frac{1}{2\pi i} \oint_{C_y} \frac{G(x_0y_0/y, y)}{y - y_0} dy$$

and, replacing y_0 by x_0y_0/r_1 in equation (9),

$$G^{\geq}(x_0, y_0) = \frac{1}{2\pi i} \oint_{C_x} \frac{G(x, x_0y_0/x)}{x - x_0} dx$$

where $|x_0| \leq r_1$ and $|y_0| \leq r_2$. The contours C_x and C_y must now lie in the annuli with radii $\{|x_0||y_0|/r_2, r_1\}$ and $\{|x_0||y_0|/r_1, r_2\}$, respectively – again restricted to ∂D_x and ∂D_y when x_0 and y_0 lie on these circles.

The matrix-transformation N_0 (corresponding to M_0 in section 2.4), representing the mapping $f(y_0) \mapsto \frac{1}{2\pi i} \oint_{C_y} \frac{f(y))}{y-y_0} dy$ for function f that is analytic in an annulus centred on the origin, is given by the following result:

Proposition 3.

(*a*) *For all $\epsilon > 0$, the integral around the contour C, defined in Fig. 1, of $f(z)/(z - z_0)$ where z_0 lies on $\partial D(0, r)$, for function $f(z)$ that has a Laurent series in an annulus containing C:*

$$\oint_C \frac{f(z)}{(z - z_0)} dz = \int_{p_2}^{p_1} \frac{f(z)}{z - z_0} dz + i(\pi f(z_0) + \epsilon(f(z_0) + 2z_0 f'(z_0)))$$

to second order in ϵ.

(*b*) *A linear interpolated vector representation of $f(z)$ is $N_0\vec{f}$, where $\vec{f}$ is a vector representation of f on ∂D at discretization points $\{z_1, \ldots, z_n\} \subset \partial D$, and the matrix N_0 is:*

$$N_0 = \frac{1}{2\pi i} \left(\tilde{K}^{(0)} + i(\pi + \epsilon) I^{n \times n} + 2z_0 \epsilon i \Delta \right)$$

with error $\Theta(n^{-2})$.

Proof. The proof for part (*a*) mirrors that of Theorem 1 with $d = 0$. The only difference is that the left hand side (LHS) is different since the $f(z_0)$ terms can no longer be collected together on the left, f not being analytic at all points in $D(0, r)$.

For part (*b*), we follow the same procedure as in Proposition 2, but here applied to the result of part (*a*). The integral term is the same as that in the proof of Proposition 2 with $d = 0$ (instead of 1). The other terms follow from Corollary 1. □

3.3. Interpolation of a partial generating function over a semi-state space

One linear mapping, with matrix-transformation $P^{\leq}$, say, is all that is needed to compute the lower partial generating function of any analytic function $G(x, y)$ – and similarly for the upper partial generating function. The calculation of $P^{\leq}$ uses Proposition 3 applied to functions f defined by $f(y) = G(r_2x_0/y, y)$ for each value of x_0 in the discretization set for x. We therefore need a mapping, call it $\Omega^{\leq}$, to yield $G(r_2x/y, y)$ from $G(x, y)$. Similarly we need the inverse mapping to go from the result, $G^{\leq}(r_2x/y, y)$ to the required $G^{\leq}(x, y)$.

3.3.1. Mapping between the discretization sets

Let the function $\omega^{\leq}(x, y) = (r_2x/y, y),\ x \in \partial D_x, y \in \partial D_y\}$. This function is represented by the $mn \times mn$ matrix $\Omega^{\leq}$ that maps the discretization points $\{(x_i, y_j) \mid 1 \leq i \leq m, 1 \leq j \leq n\}$ (ordered alphabetically) to the points $\{(r_2x_i/y_j, y_j) \mid 1 \leq i \leq m, 1 \leq j \leq n\}$. This matrix is easily defined if the discretization points are uniformly spaced around each disk and the number of them is the same for both, i.e. $m = n$, simply because in this case, any point $x_i \frac{|y_j|}{y_j}$ in ∂D_x is the point x_i with argument shifted clockwise by the argument of y_j. Thus the point is in the same set $\{x_1, \ldots, x_m\}$ since the argument-shift is a multiple of $2\pi/m$.[5] Consequently, each row and

[4] Actually, if $R_1 > r_1, R_2 > r_2$ are the largest real numbers for which $G^{(}x, y)$ is analytic in $D(0; R_1) \times D(0; R_2)$, the annuli are wider, with radii $\{r_1|y_0|/R_2, r_1\}$ and $\{r_2|x_0|/R_1, r_2\}$.

[5] When this is not the case, said point certainly lies on the x-disk boundary and interpolation may be used as an approximation.

each column in $\Omega^{\leq}$ has exactly one element equal to 1 and all the rest 0. Similar arguments apply to $\Omega^{\geq}$, the matrix-transformation of the mapping between (x_i, y_j) and $(x_i, y_j \frac{|x_i|}{x_i})$, $1 \leq i, j \leq n$.

For example, in the simplest case, if the sets of discretization points both have complex arguments $\{-\pi, 0\}$, then the 4×4 matrix representation is:

$$\Omega^{\leq} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

More concisely, the rows have their 1-elements in positions 3, 2, 1, 4.

A little more complicated, if both sets of discretization points have complex arguments $\{-\pi, -\pi/2, 0, \pi/2\}$, then the 16×16 matrix representation, $\Omega^{\leq}$, has its 1-elements in positions 9, 6, 3, 16, 13, 10, 7, 4, 1, 14, 11, 8, 5, 2, 15, 12.

Clearly, the inverse of $\Omega^{\leq}$ is equal to its transpose $(\Omega^{\leq})^T$.

3.3.2. The matrices representing partial generating functions

We can now determine the matrices $P^{\leq}, P^{\geq}$ representing lower and upper partial generating functions.

Proposition 4. *A linear interpolated vector representation of the function $G^{\leq}(r_2 x/y, y)$ is $Q^{\leq}\vec{g} + \Theta(n^{-2})$, for a certain $mn\times mn$ matrix $Q^{\leq}$, where $\vec{g}$ is a vector representation of G on $\partial(D_x \times D_y)$ at discretization point-pairs $\{(x_1, y_1), \ldots, (x_m, y_n)\} \subset \partial(D_x \times D_y)$. The matrices $Q^{\leq}$ and $Q^{\geq}$ are determined below in the proof.*

Proof. Set $x = x_i$, a discretization point on ∂D_x, for $1 \leq i \leq m$, and choose integer $k \in \{1, \ldots, n\}$. Let the discretization points on ∂D_y be $\{y_1, \ldots, y_n\}$. Identifying the points $y_{n+j} \equiv y_j$ $(1 \leq j \leq n)$, define $\theta_j = \arg y_j$ for $k+1 \leq j \leq n+k-1$, $\theta_0 = \arg y_k$, $\theta_1 = \arg y_k + \epsilon$ and $\theta_{n+1} = \arg y_k + 2\pi - \epsilon$, where $0 < \epsilon < 2\pi/n$.

We can now follow the proof of Proposition 2 with $f_j = G(r_2 x_i/y_j, y_j)$ for given i; we rename this by f_{ij} to make the dependence on i explicit. We thereby obtain the linear approximation to the integral on the right hand side of Proposition 3, corresponding to discretization point (x_i, y_k):

$$I_d = \sum_{j=1}^{n} \kappa_{ikj}(\vec{\theta}) f_{ij} = \sum_{\ell=1}^{m}\sum_{j=1}^{n} \kappa_{ikj}(\vec{\theta})\delta_{i\ell} f_{\ell j}$$

where $\delta_{i\ell}$ is the Kronecker-delta and κ_{ikj} is defined to be $\kappa^{(0)}_{1;j}$ as of equation (5) in the proof of Proposition 2 $(1 \leq i \leq m, 1 \leq j, k \leq n)$.

We now "flatten" the rank 2 tensor $f_{\ell j}$ into a vector $\vec{f}$ with its q^{th} component $(\vec{f})_q = f_{1+\lfloor \frac{q-1}{n} \rfloor, 1+\text{mod}(q-1,n)}$, according to the alphabetic ordering. Next we define the $mn \times mn$ matrix $K^{\leq}$, obtained from the rank 4 tensor $T_{ijk\ell} \equiv \kappa_{ikj}\delta_{i\ell}$ as follows:

1. Flatten the 4th and 2nd dimensions of $T_{ijk\ell}$ to obtain the rank 3 tensor $T'_{ikq} \equiv T_{ijk\ell}$ where $\ell = 1 + \lfloor \frac{q-1}{n} \rfloor$ and $j = 1 + \text{mod}(q-1, n)$;
2. Similarly, flatten the first two dimensions of T'_{ikq} to obtain the rank 2 tensor $T''_{pq} \equiv T'_{ikq}$ where $i = 1 + \lfloor \frac{p-1}{n} \rfloor$ and $k = 1 + \text{mod}(p-1, n)$
3. The matrix $K^{\leq}$ has the same elements as T''.

Then we may write $I_d = K^{\leq}\vec{f}$.

The other terms on the right hand side of Proposition 3 relate only to the y-dimension, so we can follow the proof of Proposition 2 and construct the Kronecker product with the identity matrix to obtain:

$$\left(G^{\leq}(r_2 x_i/y_j, y_j) \mid 1 \leq i \leq m, 1 \leq j \leq n\right) = Q^{\leq}\vec{f} + \Theta(n^{-2})$$

where

$$Q^{\leq} = \frac{1}{2\pi i}\left(K^{\leq} + i(\pi I^{mn\times mn} + \epsilon(I^{mn\times mn} + 2y_0 I^{m\times m} \otimes \Delta_y))\right).$$

Similarly we obtain

$$\left(G^{\geq}(x_i, r_1 y_j/x_i) \mid 1 \leq i \leq m, 1 \leq j \leq n\right) = Q^{\geq}\vec{f}' + \Theta(n^{-2})$$

where

$$Q^{\geq} = \frac{1}{2\pi i}\left(K^{\geq} + i(\pi I^{mn\times mn} + \epsilon(I^{mn\times mn} + 2x_0 \Delta_x \otimes I^{n\times n}))\right).$$

Here, $K^{\geq}$ is obtained by flattening the tensor $T_{ijk\ell}$, constructed as above but this time with n changed to m (when $m \neq n$). Note that in this case, the f_j $(1 \leq j \leq m)$ of Proposition 2 is now renamed to $f_{ij} = G(x_j, r_1 y_i/x_j)$ for each $1 \leq i \leq n$; the i and j have

swapped. Hence $\vec{f}$ is the vector obtained by flattening the rank 2 tensor f_{ij} by concatenating its columns, not its rows, i.e. now $(\vec{f})_q = f_{1+\mathrm{mod}(q-1,m),1+\lfloor\frac{q-1}{m}\rfloor}$. Consequently, the corresponding first flattening of tensor $T_{ijk\ell}$ is with respect to dimensions 2 and 4, i.e. $T'_{ikq} \equiv T_{ijk\ell}$ where $j = 1 + \lfloor\frac{q-1}{m}\rfloor$ and $\ell = 1 + \mathrm{mod}(q-1, m)$. □

Finally, we can define the matrices representing the lower and upper partial generating functions of $G(x, y)$ by using the mappings $\Omega^{\leq}$ and $\Omega^{\geq}$, respectively.

Theorem 2. *Linear interpolated vector representations of the lower and upper partial generating functions $G^{\leq}(x, y)$ and $G^{\geq}(x, y)$, respectively, of a generating function $G(x, y)$ are $g^{\leq} = P^{\leq}\vec{g}$ and $g^{\geq} = P^{\geq}\vec{g}$, where $P^{\leq} = [\Omega^{\leq}]^T \cdot Q^{\leq} \cdot \Omega^{\leq}$, $P^{\geq} = [\Omega^{\geq}]^T \cdot Q^{\geq} \cdot \Omega^{\geq}$ and $\vec{g}$ is a vector representation of G on $\partial D_x \times \partial D_y$ at discretization points $\{(x_1, y_1), (x_1, y_2), \ldots, (x_m n, y_m n)\}$ (ordered alphabetically as previously).*

4. Applications

Apart from equations that can be solved directly from the analyticity condition (see section 2 and equation (1)), all of the many previous stochastic models solved using generating functions are, to our knowledge, case-specific and intricate; the citations listed in the introduction highlight some well known examples [2,5,4,17,6,19]. The method presented here provides a technique that can be applied to diverse functional equations with few customizations.

The simple algorithm we follow is then:

Algorithm 1.

1. Discretize the x-y polydisk with appropriate radii r_1 and r_2, and with m and n points $\{x_i\}$ and $\{y_j\}$, respectively, giving nm points $\{(x_i, y_j) | 1 \leq i \leq m, 1 \leq j \leq n\}$.
2. Replace each occurrence of the unknown function $G(x, y)$ in its defining equation by its vector representation $\vec{g}$ and each operator on it by the corresponding matrix given in Table 2 that premultiplies it.
3. Collect the terms to obtain a matrix–vector equation $A\vec{g} = \vec{b}$ with known matrix A and vector $\vec{b}$.
4. Solve these equations for $\vec{g}$ by inverting the matrix A and premultiplying with $\vec{b}$. If the equations are homogeneous ($\vec{b} = \vec{0}$), as in the case where G is a probability generating function, one row is replaced by another that represents the normalizing equation, e.g. if $G(1, 1) = 1$, as we will find in the JSQ-application (Section 4.3), a row is replaced with one containing all 1's and the corresponding component of $\vec{b}$ is changed from 0 to 1.
   The solution, $A^{-1}\vec{b}$, computes the vector of values of the function $G(x, y)$ at the discretization points lying on the distinguished boundary of the polydisk – call it $\vec{g}_0$.
5. The function $G(x, y)$ is then approximated at all points on the polydisk by interpolation (linear or higher order), $G_0 = \mathfrak{I}(\vec{g}_0)$.

We consider response time distribution numerically in three distinct types of two-dimensional Markov models: a tandem pair of first-come-first-served (FCFS) queues with negative customers; a tandem pair of queues with processor-sharing queueing discipline (PS) at the first queue and FCFS at the second; and a pair of parallel queues with join-shortest-queue scheduling (JSQ). All three models allow some form of overtaking [20], i.e. the possibility that an arriving customer leaves the network *after* a later arrival, due to:

1. *never* leaving the network because of being removed by a negative customer;
2. the PS queueing discipline at the first queue, where every customer in service has the same chance of being the next departure;
3. a later JSQ-customer joining the other queue that turns out to be "faster".

In each application, the first four moments of response time are computed by numerical solution of the functional equation, as is the probability density function, which is obtained by numerical inversion of its Laplace transform and order-3 interpolation. The results are compared with regenerative simulations that each run for 500 000 regeneration cycles; see [10]. Following [3], these sets of cycles are partitioned into batches of size equal to approximately the square root of the number of cycles, i.e. 707, giving a total of 707 batches as well. The moments are estimated from the complete set of simulated response times and the probability density function (pdf) from the corresponding relative frequency histogram. 95% confidence bands (CBs) are computed by the batch means method – for each moment and each pdf-point – using the sample variance of each batch's mean value.

4.1. Tandem pair of FCFS queues with negative customers

The first application we consider investigates response time density in a two-queue tandem network with negative customers that remove regular customers from a queue [8,9]. This network is depicted in Fig. 2.

In terms of the functional equation defining the solution, this turns out to be of the same type as the one arising from a three-node path in a network with a "short-cut" that allows literal overtaking, considered in [6]. The parameters are as follows: for queue $i \in \{1, 2\}$, the regular external arrival rate is λ_i^+, the negative external arrival rate is λ_i^-, and the service rate is μ_i. All customers not

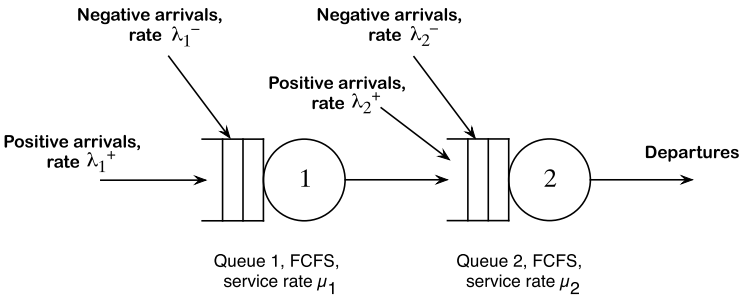


Fig. 2. Application 1: Tandem pair of M/M/1 G-queues.

deleted by a negative customer pass to queue 2 on leaving queue 1; customers completing service at queue 2 exit the network and the time elapsed between a customer's arrival at queue 1 and leaving queue 2 is its response time.

For this model, the Laplace transform of sojourn time density, jointly with the probability of a customer not being killed, may be computed as $W^*(s) = (1-\rho_1)(1-\rho_2)\frac{\mu_1}{\lambda_1^+} y_1(s) G(\rho_2, 0, y_1(s), s)$ for a certain known expression $y_1(s)$, where $\rho_1 = \frac{\lambda_1^+}{\lambda_1^- + \mu_1}$ and $\rho_2 = \frac{\mu_1\rho_1 + \lambda_2^+}{\lambda_2^- + \mu_2}$. The generating function $G(x, y, z, s)$ is given by the following functional equation:

$$R(x,y,z,s)G(x,y,z,s) = \left(\lambda_1^- + \mu_1 - \frac{\lambda_1^+}{z}\right) G(x,y,0,s) - \frac{\mu_1 z + \lambda_2^+}{y} G(x,0,z,s) + \frac{\mu_2}{(1-y)(1-z)} \tag{10}$$

where

$$R(x,y,z,s) = s + \lambda_1 + \lambda_2 + \mu_1 + \mu_2(1-x) - \frac{\lambda_1^+}{z} - \lambda_1^- z - \frac{\lambda_2^+}{y} - \lambda_2^- y - \mu_1 \frac{z}{y}.$$

This is an instance of the generic equation (2), and it is in this sense that the queuing system can be considered isomorphic to that of [6]. Both papers [6,12] used boundary value methods to clearly define the solution of their respective problem and showed how to compute the required Laplace transform. Unfortunately, in neither case are there any numerical results available. However, here we simply compute the Laplace transform, at given s and for the required value of x, by following the steps described in Algorithm 1. Henceforth, when the meaning is clear, we omit the x- and s-arguments, which are the same for each instance of G in the equation, treating them as parameters.

The Laplace transform $W^*(s)$ may now be calculated at chosen points s and then inverted to give the probability density function of response time $w(t)$ for $t \geq 0$. We used the Fixed Talbot inversion algorithm[6] [1]. To calculate the Laplace transform of the sojourn time density of a customer *not killed*, we divide the expression for the Laplace transform given above by $W^*(0)$.

4.1.1. Numerics

Numerically, it is more convenient to work with the function $H(x,y,z,s) = (1-y)(1-z)G(x,y,z,s)$ since, by construction, we know that when $s = 0$, $G(x,y,z,0) \propto \frac{1}{(1-y)(1-z)}$ (see [12]) and so $H(x,y,z,0)$ is constant. The corresponding equation satisfied by H is then:

$$\begin{aligned} &R(x,y,z,s)yzH(x,y,z,s) - ((\lambda_1^- + \mu_1)z - \lambda_1^+)(1-z)yH(x,y,0,s) \\ &\qquad + (\mu_1 z + \lambda_2^+)(1-y)zH(x,0,z,s) = \mu_2 yz \end{aligned} \tag{11}$$

The Laplace transform of the equilibrium sojourn time density function may now be expressed as

$$W^*(s) = \frac{(1-\rho_1)(1-\rho_2)\mu_1}{\lambda_1^+} \left(\frac{y_1(s)}{1-y_1(s)}\right) H(\rho_2, 0, y_1(s), s)$$

by virtue of the product form of the initial state probability at equilibrium. This may be evaluated for any value of $s \geq 0$ via the generalized Cauchy integral of $H(y,z)$ over the distinguished boundary of the polydisk with $(0, y_1(s))$ as the interior point.

Applying Algorithm 1 to equation (11) and using Table 2, we obtain the matrix–vector equation[7]:

$$\Big(\Lambda(\vec{a}_1) - \Lambda(\vec{a}_2)V_0 + \Lambda(\vec{a}_3)U_0\Big)\vec{h} = \Lambda(\vec{a}_4) \tag{12}$$

where $\vec{a}_1, \vec{a}_2, \vec{a}_3, \vec{a}_4$ are the vector representations of the functions A_1, A_2, A_3, A_4 defined by

[6] See https://mpmath.org/doc/current/calculus/inverselaplace.html.

[7] We re-label the variables x, y used in Algorithm 1 by $y \mapsto z$, $x \mapsto y$, to be compatible with the notation of equation (10) and the paper [12].

$$A_1(y,z) = R(x,y,z,s)yz;$$
$$A_2(y,z) = ((\lambda_1^- + \mu_1)z - \lambda_1^+)(1-z)y;$$
$$A_3(y,z) = (\mu_1 z + \lambda_2^+)(1-y)z;$$
$$A_4(y,z) = \mu_2 yz \text{ for all } x, y.$$

Solving this equation we obtain the solution-estimate H_0 for H as the linear interpolation of the solution vector $\vec{h}$.

For interest, we compared the results obtained by our method and by the BVP method of [12] in Fig. 9 (see Appendix 6.3). Not surprisingly, the curves are very close.

4.1.2. Moments

The nth moment of response time is $(-1)^n$ multiplied by the nth derivative of $W^*(s)$ evaluated at $s = 0$. This requires the partial derivatives of $H(x,y,z,s)$ with respect to (wrt) s at $s = 0$, which are determined as follows.

First, note that the solution for the vector representation of $H(x_0,y,z,s)$ at given x_0 in equation (11), with RHS changed to $r(x,y,z,s)$, is $\Gamma\vec{v}$, where $\Gamma = \left(\Lambda(\vec{a}_1) - \Lambda(\vec{a}_2)V_0 + \Lambda(\vec{a}_3)U_0\right)^{-1}$ and the column vector $\vec{v} = (r(x,y_i,z_j,s) \mid 1 \le i \le m, 1 \le j \le n)^T$, ordered alphabetically; so in the above, we had $r(x,y,z,s) = \mu_2 yz$ giving $\vec{h} = \Gamma(\mu_2 y_i z_j \mid 1 \le i \le m, 1 \le j \le n)^T$.

Lemma 3. *The kth partial derivative of $H(x_0,y,z,s)$ wrt s at $s = 0$, where $x_0 \in D_x$ is fixed, has vector representation $\vec{h}_k$ given by the recurrence:*

$$\vec{h}_0 = \Gamma(\mu_2 y_i z_j \mid 1 \le i \le m, 1 \le j \le n)^T$$
$$\vec{h}_n = -n\Gamma\vec{h}_{n-1} \text{ for } n \ge 1.$$

Proof. Differentiating equation (11) $n \ge 1$ times wrt s yields, using Leibnitz' result for multiple differentiation of a product and noting that $R^{(1)}(y,z) = 1$,

$$R(x_0,y,z,s)H^{(n)}(y,z) - (\lambda_1^- + \mu_1 - \lambda_1^+/z)(1-z)H^{(n)}(x_0,y,0,s) + \frac{\mu_1 z + \lambda_2^+}{y}(1-y)H^{(n)}(x_0,0,z,s) = -n\,H^{(n-1)}(y,z)$$

where the parenthesized superscripts denote multiple differentiation wrt s.

The LHS of this equation is identical to the LHS of equation (11) with H replaced by $H^{(n)}$. The RHS is $-nH^{(n-1)}(y,z)$ and so the same method of solution can be used to compute (at $s = 0$) the vector representation of $H^{(n)}(y,z)$ by changing the RHS of equation (11) to $-n\vec{h}_{n-1}$, as required. □

Notice that there is no need to solve any more functional equations; the same inverse matrix $\left(\Lambda(\vec{a}_1) - \Lambda(\vec{a}_2)V_0 + \Lambda(\vec{a}_3)U_0\right)^{-1}$ is all that is needed for each derivative; only the RHS $-nH^{(n-1)}(y,z)$ changes.

To complete the calculation of the nth moment, we need to differentiate, an arbitrary number of times, the product $\left(\frac{y_1(s)}{1-y_1(s)}\right) H(\rho_2,0,y_1(s),s)$, which we again do by Leibnitz' result. Higher derivatives of the term $\frac{y_1(s)}{1-y_1(s)}$ are straightforward to compute using standard mathematical software, but it is more complex for the expression $H(\rho_2,0,y_1(s),s)$ since s occurs in two of its parameters, requiring partial derivatives wrt z and u of $H(x,y,z,u)$, as well as derivatives of $y_1(s)$ wrt s.

We used mathematical software (Mathematica, version 12.3.1 [15]) to compute symbolic expressions for the first four derivatives wrt s of the expression $\alpha(s)\beta(\gamma(s),s)$ for function-variables α,β,γ. Then, substituting $\alpha(s) = \frac{y_1(s)}{1-y_1(s)}$, $\beta(z,s) = H(\rho_2,0,y_1(s),s)$, and $\gamma(s) = y_1(s)$ gives the results we need, apart from the derivatives of H wrt s and z at $s = 0$. The former are given by Lemma 3 and the latter are computed by Cauchy integrals of H and its partial derivatives wrt s over the boundary of the polydisk with interior point $(0, y_1(0))$.

4.1.3. Comparison with simulation

We computed the linear interpolation of the function H using a polydisk with $m = n = 64$ points around each circle in the complex y- and z-planes. We then used the recurrence of Lemma 3 to calculate the first four moments of response time and compared the results with those obtained from the regenerative simulation in Table 3.

We note that all the computed moments lie within the 95% confidence bands, except for the mean which is fractionally outside the lower band, and the percentage errors range from 1 to 3%, even for the fourth moment. This gives strong support for the accuracy of the model.

Next we plotted the probability density function $w(t)$ by numerically inverting the Laplace transform $W^*(s)$ as described above (just before Section 4.1.1). For each point t, we used 16 values of the Laplace parameter s in the Fixed Talbot inversion algorithm [1], requiring $16n$ solutions for the generating function when n t-points are required to interpolate the density function with sufficient

Table 3
Response time moments for a customer not killed in the two-server tandem network, where the two queues are FCFS-RCE M/M/1 ±-queues with parameters $\lambda_1^+ = 3$, $\lambda_1^- = 2$, $\lambda_2^+ = 0$, $\lambda_2^- = 2$, $\mu_1 = 2$, $\mu_2 = 1$: comparison with simulation including 95% confidence bands (CBs).

Moment	Model	Simulation	95% CB
1st	2.74346	2.76008	± 0.0163813
2nd	12.9699	13.1498	± 0.19436
3rd	87.6995	89.6254	± 2.62747
4th	771.309	792.416	± 41.5083

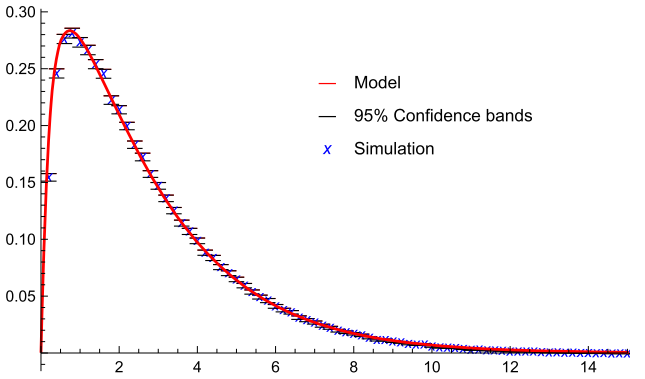


Fig. 3. Response time density ($w(t)$) for a customer not killed in the two-server tandem network, where the two queues are FCFS-RCE M/M/1 ±-queues with parameters $\lambda_1^+ = 3$, $\lambda_1^- = 2$, $\lambda_2^+ = 0$, $\lambda_2^- = 2$, $\mu_1 = 2$, $\mu_2 = 1$: comparison with simulation including 95% confidence bands (CBs).

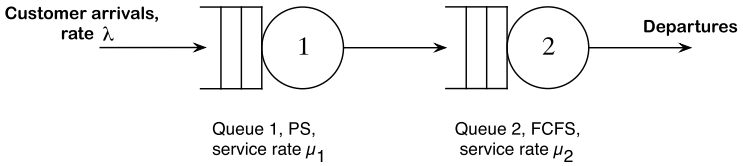


Fig. 4. Application 2: Tandem pair of PS and M/M/1 queues.

accuracy. Typically n is in the range 10–20. The resulting probability density function is shown in Fig. 3, where we superimpose the simulation-generated relative frequency histogram with 95% confidence bands.

Again we see a high degree of accuracy; clear by "eyeballing" but also with most of the model's points lying inside the confidence bands.

4.2. Tandem pair of PS and FCFS queues

The second application we consider investigates response time distribution in a two-server tandem network with PS at the first server and FCFS at the second – see Fig. 4. The arrival process at node 1 is Poisson with rate parameter λ and nodes 1 and 2 have independent exponential service times with parameters μ_1 and μ_2, respectively.

The Laplace transform of the sojourn time density function is given in [4] as $W^*(s) = (1-\rho_1)(1-\rho_2)G(\rho_1, \rho_2, s)$, where $\rho_1 = \frac{\lambda}{\mu_1}$, $\rho_2 = \frac{\lambda}{\mu_2}$ and $G(x, y, s)$ is the generating function given by the following functional equation:

$$G(x,y,s)\left(\lambda + \mu_1(1-\frac{x}{y}) + \mu_2(1-y) + s\right) + \frac{\partial G(x,y,s)}{\partial x}\left(\lambda(x-1) + \mu_1 x(1-\frac{x}{y}) + \mu_2 x(1-y) + xs\right)$$
$$= G(x,0,s)\left(\mu_2 - \mu_1\frac{x}{y}\right) + \frac{\partial G(x,0,s)}{\partial x}x\left(\mu_2 - \mu_1\frac{x}{y}\right) + \frac{\mu_1\mu_2}{(1-x)(\mu_2(1-y)+s)} \quad (13)$$

This functional equation is more complicated than the one in the previous section in that it contains partial derivatives – here with respect to only one of the variables, x. Numerically, it is again more convenient to work with the function $H(x, y, s)$ which is defined as $H(x, y, s) = (1-x)(1-y)G(x, y, s)$. Then, by the definition in [4] in terms of conditional Laplace transforms, when $s = 0$, $H(x, y, 0) = 1$ for all x, y. The corresponding equation satisfied by H is

$$H(x,y,s)\left(\mu_2 y+\frac{sy-\mu_1 x+\mu_1 y}{(1-y)}\right)-\frac{\partial H(x,y,s)}{\partial x}\left(\frac{(1-x)\left[x(x-y)\mu_1+y(\lambda-x(s+\lambda)-x(1-y)\mu_2)\right]}{1-y}\right)$$
$$+H(x,0,s)\left(\mu_1 x-\mu_2 y\right)+\frac{\partial H(x,0,s)}{\partial x}(1-x)x\left(\mu_1 x-\mu_2 y\right)=\frac{\mu_1\mu_2(1-x)y}{s+\mu_2(1-y)} \tag{14}$$

4.2.1. Numerics

The Laplace transform of the sojourn time density function may now be expressed as $W^*(s)=H(\rho_1,\rho_2,s)$. The calculation of $H(x,y)$ at the chosen discretization point-pairs, represented by the alphabetically ordered vector $\vec{h}=(H(x_i,y_j)|1\le i\le m,1\le j\le n)$, is then according to Algorithm 1, using Table 2. We obtain the matrix–vector equation:

$$\left(\Lambda(\vec{a}_1)+\Lambda(\vec{a}_2)V_0-\Lambda(\vec{a}_3)M_{1,0}+\Lambda(\vec{a}_4)M_{1,0}V_0\right)\vec{h}=\Lambda(\vec{a}_5) \tag{15}$$

where $\vec{a}_1,\vec{a}_2,\vec{a}_3,\vec{a}_4,\vec{a}_5$ are the vector representations of the functions A_1,A_2,A_3,A_4,A_5 defined by

$$A_1(x,y)=\mu_2 y+\frac{sy-\mu_1 x+\mu_1 y}{(1-y)};$$
$$A_2(x,y)=\mu_1 x-\mu_2 y;$$
$$A_3(x,y)=\frac{(1-x)\left[x(x-y)\mu_1+y(\lambda-x(s+\lambda)-x(1-y)\mu_2)\right]}{1-y};$$
$$A_4(x,y)=(1-x)x\left(\mu_1 x-\mu_2 y\right);$$
$$A_5(x,y)=\frac{\mu_1\mu_2(1-x)y}{s+\mu_2(1-y)}.$$

Solving this equation we obtain the solution-estimate H_0 for H as the linear interpolation of the solution vector $\vec{h}$.

4.2.2. Moments

Moments are computed by the method used in the previous application to determine the derivatives wrt s of $H(x,y,s)$. It is a more complicated procedure here because of the derivatives wrt x in the defining equation (14) but, once computed, the kth moment is simply $(-1)^{(k)}H^{(k)}(x,y,0)$. We use the following lemma, analogous to Lemma 3.

Lemma 4. *The kth partial derivative of $H(x,y,s)$ wrt s at $s=0$ has vector representation $\vec{h}_k$ given by the recurrence:*

$$\vec{h}_0=(1,\ldots,1)^T,$$
$$\vec{h}_k=\Gamma\left(-k\left(\frac{y}{1-y}\right)\vec{h}_{k-1}-k\left(\frac{(1-x)xy}{1-y}\right)h'_{k-1}+(-1)^k k!\vec{b}_k\right),$$
$$h'_k=M_{1,0}\vec{h}_k$$

where $\Gamma=\left(\Lambda(\vec{a}_1)+\Lambda(\vec{a}_2)V_0-\Lambda(\vec{a}_3)M_{1,0}+\Lambda(\vec{a}_4)M_{1,0}V_0\right)^{-1}$ *and* $\vec{b}_k=(\frac{\mu_1\mu_2(1-x)y}{(\mu_2(1-y))^{k+1}}\,|\,1\le i\le m,1\le j\le n)^T$ *ordered alphabetically, for* $k\ge 0$.

Proof. First, as already noted, $\vec{h}_0=H(x,y,0)=1$ by construction. Differentiating equation (14) k times wrt s yields, using Leibnitz' result for multiple differentiation of a product, for $k\ge 1$,

$$H^{(k)}(x,y)\left(\mu_2 y+\frac{sy-\mu_1 x+\mu_1 y}{(1-y)}\right)-\frac{\partial H^{(k)}(x,y)}{\partial x}\left(\frac{(1-x)\left[x(x-y)\mu_1+y(\lambda-x(s+\lambda)-x(1-y)\mu_2)\right]}{1-y}\right)$$
$$+H^{(k)}(x,0)\left(\mu_1 x-\mu_2 y\right)+\frac{\partial H^{(k)}(x,0)}{\partial x}(1-x)x\left(\mu_1 x-\mu_2 y\right)+kH^{(k-1)}(x,y)\left(\frac{y}{1-y}\right)$$
$$+k\frac{\partial H^{(k-1)}(x,y)}{\partial x}\left(\frac{(1-x)xy}{1-y}\right)=(-1)^k k!\frac{\mu_1\mu_2(1-x)y}{(s+\mu_2(1-y))^{k+1}}$$

The first four terms have exactly the same coefficients as those of the H-terms in equation (14). Hence the same method of solution as we used to compute $H(x,y)$ can be used to compute $H^{(k)}(x,y)$ if we change the RHS to

$$r_k(x,y,s)=-kH^{(k-1)}(x,y)\left(\frac{y}{1-y}\right)-k\frac{\partial H^{(k-1)}(x,y)}{\partial x}\left(\frac{(1-x)xy}{1-y}\right)+(-1)^k k!\frac{\mu_1\mu_2(1-x)y}{(s+\mu_2(1-y))^{k+1}}$$

Setting $s=0$ then completes the proof. □

Again we note that there is no need to recalculate the inverse of a matrix for each moment, the same inverse matrix Γ is all that is needed.

Table 4
Response time moments for a two-server tandem network with PS at the first server and FCFS at the second and parameters $\lambda = 1$, $\mu_1 = 2$, $\mu_2 = 3$: comparison with simulation.

Moment	Model	Simulation	95% CB
1st	1.49999	1.50011	± 0.00523963
2nd	4.25421	4.25715	± 0.0453722
3rd	20.5703	20.6998	± 0.53194
4th	149.59	156.799	± 8.48684

Table 5
Response time moments for a two-server tandem network with PS at the first server and FCFS at the second and parameters $\lambda = 1$, $\mu_1 = 1.4$, $\mu_2 = 1.5$: comparison with simulation.

Moment	Model	Simulation	95% CB
1st	4.49954	4.49642	± 0.0169931
2nd	38.5328	38.2871	± 0.413403
3rd	574.181	565.849	± 15.236
4th	13 833.0	13 623.6	± 899.383

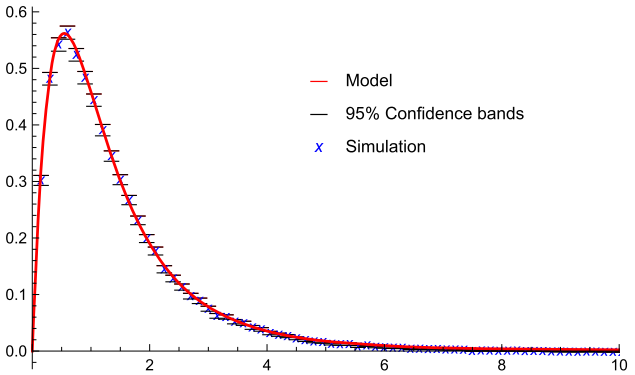


Fig. 5. Response time density for a two-server tandem network with PS at the first server and FCFS at the second and parameters $\lambda = 1$, $\mu_1 = 2$, $\mu_2 = 3$.

4.2.3. Comparison with simulation

We again computed the linear interpolation of the function H using a polydisk with $m = n = 125$ points around each circle in the complex x- and y-planes. We then used the recurrence of Lemma 4 to calculate the first four moments of response time and compared the results with those obtained from the corresponding regenerative simulation, first in Table 4 for a two-server tandem network with parameters $\lambda = 1$, $\mu_1 = 2$, $\mu_2 = 3$.

Although there are no published results to compare with (to our best knowledge), we know that mean response time must be 1.5 since this is the same whether the first server has PS or FCFS queueing discipline; so the error of less than 0.1 percent is encouraging. We also note that all the moments lie comfortably inside the 95% confidence bands of the simulation, with percentage errors of less than 0.1%, 0.2%, 2% and 5% for the first four moments, respectively.

It is interesting to note the corresponding results for two FCFS queues with the same parameters. These are known to be independent at equilibrium so their response time moments are easily calculated as $1.5, 3.5, 11.25, 46.5$ for the first four. Our results show the variance is significantly increased by the PS at queue 1, and higher moments are increasingly greater.

This network parameterization gives rather low utilizations of 0.5 and 0.333 at servers 1 and 2 respectively, so good accuracy might be expected. We therefore computed corresponding results for a network with parameters $\lambda = 1$, $\mu_1 = 1.4$, $\mu_2 = 1.5$, giving utilizations 0.714 and 0.666. The results are shown in Table 5, where we still see excellent accuracy – actually improved accuracy in all but the first moment, with all results of the model lying near the middle of their confidence bands.

The first four moments in the corresponding tandem pair of FCFS queues are 4.5, 30.5, 276.75 and 3151.5, showing somewhat bigger percentage differences from the PS–FCFS network at higher moments than in the lower utilization model, as might be expected.

Finally, we plotted the probability density function $w(t)$ by numerically inverting the Laplace transform $W^*(s)$ as in the previous application. Because the calculations are more computationally expensive, requiring the Laplace transform at 32 points for each value of t chosen in the interpolation for $w(t)$, we used a polydisk with only $m = n = 64$ points around each circle in the complex x- and y-planes. Figs. 5 and 6 show the graphs for the same two parameterizations, with the corresponding simulation results superimposed, along with 95% confidence bands. Again we see a high degree of accuracy in both graphs, with most of the model's points lying inside the confidence bands.

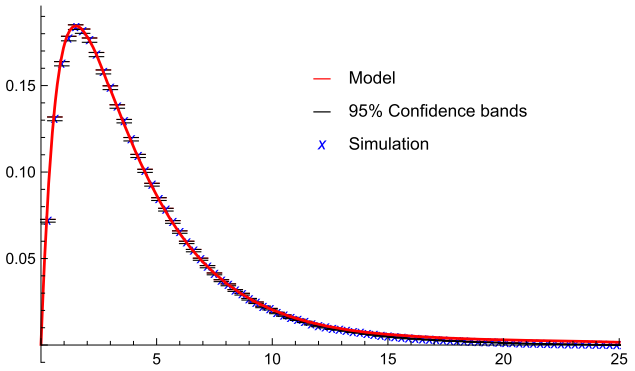


Fig. 6. Response time density for a two-server tandem network with PS at the first server and FCFS at the second and parameters $\lambda = 1$, $\mu_1 = 1.4$, $\mu_2 = 1.5$.

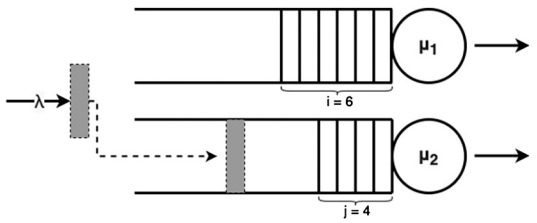


Fig. 7. Application 3: Join-the-shortest-queue scheduling with two queues.

4.3. JSQ scheduling with two parallel queues

Consider now the following scheduling problem involving two queues and a single server. Poisson arrivals occur at rate λ and join the shorter of the two queues, or, if the lengths are equal, join queue 1 with probability a_1 and queue 2 with probability $a_2 = 1 - a_1$. Each queue has an independent server with exponential service times, parameters μ_1, μ_2.[8] Such a network is depicted in Fig. 7, where a new arrival finds both servers busy with 6 customers waiting in queue 1 and 4 waiting in queue 2, and so joins queue 2.

This apparently simple problem was first considered by Kendall in the symmetric case [16], where $a_1 = a_2 = 1/2$ and $\mu_1 = \mu_2$. Subsequently, Flatto and McKean obtained a complex closed form solution for this case in [7].

At time $t \geq 0$, let $N_1(t), N_2(t)$ be the queue length random variables at queues $1, 2$ respectively, and $\pi_{i,j}(t) = \mathbb{P}(N_1(t) = i, N_2(t) = j)$ for integers $i, j \geq 0$. Defining $\pi_{i,j} = \lim_{t\to\infty} \pi_{i,j}(t)$, at equilibrium (for now assuming this exists), let the probability generating function $G(x,y) = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \pi_{i,j} x^i y^j$. This is given by the functional equation in the following result.

Theorem 3. *The probability generating function G satisfies the following functional equation, for all complex values x, y with $|x| \leq 1$ and $|y| \leq 1$:*

$$\begin{aligned}\Big(\lambda(1 - a_2 x - a_1 y) + \mu_1(1 - 1/x) + \mu_2(1 - 1/y)\Big)G(x,y) = \\ a_1\lambda(x - y)G^{\leq}(x,y) + a_2\lambda(y - x)G^{\geq}(x,y) \\ + \mu_1(1 - 1/x)G(0,y) + \mu_2(1 - 1/y)G(x,0)\end{aligned} \tag{16}$$

where $G^{\geq}(x,y) = \sum_{i\geq j\geq 0} \pi_{i,j} x^i y^j = \sum_{i=0}^{\infty}\sum_{j=0}^{i} \pi_{i,j} x^i y^j$ *and* $G^{\leq}(x,y) = \sum_{0\leq i\leq j} \pi_{i,j} x^i y^j = \sum_{i=0}^{\infty}\sum_{j=i}^{\infty} \pi_{i,j} x^i y^j$.

Proof. At equilibrium, the balance equations of the Markov process defining the JSQ model are:

$$\begin{aligned}(\lambda + \mu_1 I_{i>0} + \mu_2 I_{j>0})\pi_{i,j} = \lambda\pi_{i-1,j} I_{0<i<j+1} + \lambda\pi_{i,j-1} I_{0<j<i+1} + \\ \lambda(a_1\pi_{i-1,j} I_{i=j+1} + a_2\pi_{i,j-1} I_{j=i+1}) + \\ \mu_1\pi_{i+1,j} + \mu_2\pi_{i,j+1}\end{aligned} \tag{17}$$

[8] An analogous load-balancing problem may be solved similarly, where there are independent Poisson arrival streams with rates λ_1, λ_2 and a single server of rate μ that picks from the front of the longer of the two queues when starting a new service period, or either one with specified probability when the queue lengths are equal and positive.

where I is the indicator function, $I_{\text{true}} = 1$ and $I_{\text{false}} = 0$. Multiplying equation (17) throughout by $x^i y^j$ and summing over $i, j \geq 0$ yields, after some simplification

$$
\begin{aligned}
&(\lambda + \mu_1 + \mu_2)G(x,y) - \mu_1 \sum_{j=0}^{\infty} \pi_{0,j} y^j - \mu_2 \sum_{i=0}^{\infty} \pi_{i,0} x^i \\
&\quad = \lambda x \sum_{0 \leq i < j} \pi_{i,j} x^i y^j + \lambda y \sum_{0 \leq j < i} \pi_{i,j} x^i y^j + \lambda a_1 x \sum_{j=0}^{\infty} \pi_{j,j} x^j y^j + \lambda a_2 y \sum_{i=0}^{\infty} \pi_{i,i} x^i y^i \\
&\quad + \mu_1 / x \Big(G(x,y) - \sum_{j=0}^{\infty} \pi_{0,j} y^j \Big) + \mu_2 / y \Big(G(x,y) - \sum_{i=0}^{\infty} \pi_{i,0} x^i \Big) \\
&\quad = \lambda x G^{<}(x,y) + \lambda y G^{>}(x,y) - \lambda(1-a_1) x \sum_{j=0}^{\infty} \pi_{j,j} x^j y^j - \lambda(1-a_2) y \sum_{i=0}^{\infty} \pi_{i,i} x^i y^i \\
&\quad + \mu_1 / x \Big(G(x,y) - G(0,y) \Big) + \mu_2 / y \Big(G(x,y) - G(x,0) \Big)
\end{aligned}
$$

Noting that

$$\sum_{j=0}^{\infty} \pi_{j,j} x^j y^j = G^{>}(x,y) + G^{<}(x,y) - G(x,y)$$

we obtain equation (16). □

4.3.1. Numerics

Applying Algorithm 1 to equation (16) and using Table 2, we obtain the matrix–vector equation:

$$\Big(\Lambda(\vec{a}_1) - \Lambda(\vec{a}_2) P^{<} - \Lambda(\vec{a}_3) P^{>} - \Lambda(\vec{a}_4) U_0 - \Lambda(\vec{a}_5) V_0 \Big) \vec{g} = (0, \ldots, 0)^T \tag{18}$$

for the vector representation of the pgf G on $D_x \times D_y$, where $\vec{a}_1, \vec{a}_2, \vec{a}_3, \vec{a}_4$ are the vector representations of the functions A_1, A_2, A_3, A_4 defined by

$$
\begin{aligned}
A_1(y,z) &= \lambda(1 - a_2 x - a_1 y) + \mu_1(1 - 1/x) + \mu_2(1 - 1/y); \\
A_2(y,z) &= a_1 \lambda(x - y); \\
A_3(y,z) &= a_2 \lambda(y - x); \\
A_4(y,z) &= \mu_1(1 - 1/x); \\
A_5(y,z) &= \mu_2(1 - 1/y).
\end{aligned}
$$

This equation is homogeneous with unique solution only up to a multiplicative constant. The unique solution is determined by the normalizing condition $G(1,1) = 1$.

Now, a pgf is usually analytic in circles of radius greater than one – e.g. ρ^{-1} in the case of an M/M/1 queue with utilization ρ. Here we would expect it to have radii of convergence $(\mu_1 + \mu_2)/\lambda$, so we used radii r_1, r_2 between 1 and this value to define the polydisk. We can then approximate the unique solution vector representation $\vec{g}$ by replacing one of the rows of the matrix $\Lambda(\vec{a}_1) - \Lambda(\vec{a}_2) P^{<} - \Lambda(\vec{a}_3) P^{>} - \Lambda(\vec{a}_4) U_0 - \Lambda(\vec{a}_5) V_0$ – row i, say, $1 \leq i \leq nm$ – by one of the rows of the matrix $U_1 V_1$, for which $U_1 V_1 \vec{g}$ approximates $G(1,1)$. The zero-vector on the RHS is then replaced by $\vec{e}_i^T$, the i^{th} unit vector in the standard basis (i.e. the column vector with 1 for its i^{th} component and the others all 0). We thereby obtain the solution-estimate G_0 for G as the linear interpolation of this solution vector, $\vec{g}_0$. We name the inverse of the modified matrix Γ, consistent with the previous two applications, so the equation we solve is $\Gamma \vec{g} = \vec{e}_i$.

4.3.2. Queue length moments

From the pgf of the joint queue length distribution, $G_0(x.y)$, the p^{th} factorial moments can be found by estimating derivatives of the pgf at the point $(1,1)$, i.e. $U_1 V_1 M_{p,0} \vec{g}$ and $U_1 V_1 M_{0,p} \vec{g}_0$ for the vector representation of the p^{th} factorial moment of the lengths of queue 1 and queue 2, respectively. Furthermore, the covariance of the queue lengths is simply approximated by $U_1 V_1 M_{1,1} \vec{g}_0$.[9]

4.3.3. Laplace transform of response time density

Next we used the pgf to obtain the response time density's Laplace transform $W^*(s)$ as follows, assuming the queueing discipline is FCFS in each queue.

[9] Of course all of these quantities are vector representations for constant functions, so their components are all the same. Without loss of generality, then, we take the first component.

Table 6
Queue length moments for the JSQ model with parameters $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$: comparison with simulation.

Moment	Model	Simulation	95% CB
1st	3.3265	3.31212	± 0.0199906
2nd	19.5258	19.4012	± 0.29771
3rd	163.348	161.994	± 4.89695
4th	1 795.62	1 769.13	± 93.7394

(a) Queue length moments at queue 1

Moment	Model	Simulation	95% CB
1st	2.76837	2.76441	±0.0190922
2nd	15.6601	15.6946	±0.268722
3rd	128.448	129.695	±4.322
4th	1 384.94	1 410.79	±81.6966

(b) Queue length moments at queue 2

Table 7
Response time moments for the JSQ model with parameters $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$: comparison with simulation.

Moment	Model	Simulation	95% CB
1st	6.07415	6.10602	± 0.0370392
2nd	79.3979	80.5682	± 1.20298
3rd	1 678.18	1 725.23	± 49.2549
4th	49 942.	52 009.3	± 2 592.42

Proposition 5. *The Laplace transform of response time in the pair of M/M/1 queues with JSQ scheduling is*

$$W^*(s) = a_2\left(\frac{\mu_1}{s+\mu_1}\right)[G(\mu_1/(s+\mu_1),1) - G^{\geq}(\mu_1/(s+\mu_1),1)] + a_1\left(\frac{\mu_1}{s+\mu_1}\right)G^{\leq}(\mu_1/(s+\mu_1),1)$$
$$+ a_1\left(\frac{\mu_2}{s+\mu_2}\right)[G(1,\mu_2/(s+\mu_2)) - G^{\leq}(1,\mu_2/(s+\mu_2))] + a_2\left(\frac{\mu_2}{s+\mu_2}\right)G^{\geq}(1,\mu_2/(s+\mu_2))$$

Proof. See Appendix 6.4. □

This leads to the following approximation to $W^*(s)$ for any given s in terms of vector representations and the matrices of Table 2:

$$W^*(s) \approx \left(a_2 s_1[U_{s_1}V_1 - U_{s_1}V_1P^{\geq}] + a_1 s_1 U_{s_1}V_1P^{\leq} + a_1 s_2[U_1V_{s_2} - U_1V_{s_2}P^{\leq}] + a_2 s_2 U_1V_{s_2}P^{\geq}\right)\vec{g}_0 \tag{19}$$

where $s_1 = \mu_1/(s+\mu_1)$ and $s_2 = \mu_2/(s+\mu_2)$. The density function of response time is obtained by inverting its Laplace transform numerically, requiring approximations to $W^*(s)$ at several values of s, calculated by approximation (19).

4.3.4. Moments of response time

Proposition 5 can be used to compute response time moments by differentiation of $W^*(s)$ at $s = 0$. This is straightforward but somewhat tedious because $W^*(s)$ is a sum of four products of expressions in s, requiring Leibnitz' result for each. The derivatives wrt s of the generating function $G(\mu_1/(s+\mu_1),1)$ are first computed symbolically in terms of the partial derivatives of $G(x,y)$ wrt x, y at $x = y = 1$ (corresponding to $s = 0$ in $\mu_1/(s+\mu_1)$). These results have already been obtained in the calculation of the vector representations of the derivatives of $G(x,y)$, from which the queue length moments followed; see section 4.3.2.

In addition, we have to evaluate derivatives of the partial generating functions $G^{\geq}$ and $G^{\leq}$. Their vector representations are simply obtained using the matrices $P^{\leq}$ and $P^{\geq}$, giving $\vec{g}^{\leq}_{px} = U_1V_1M_{p,0}P^{\leq}\vec{g}_0$, $g^{\geq}_{px} = U_1V_1M_{p,0}P^{\geq}\vec{g}_0$, $g^{\leq}_{py} = U_1V_1M_{0,p}P^{\leq}\vec{g}_0$ and $g^{\geq}_{py} = U_1V_1M_{0,p}P^{\geq}\vec{g}_0$, for the vector representations of the corresponding p^{th} derivatives wrt x and y at $x = y = 1$.

4.3.5. Comparison with simulation

We tested our model on a pair of parallel JSQ servers with the following parameters: $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$. With this choice of parameters, either queue would be saturated on its own since $\lambda > \mu_2 > \mu_1$, so the scheduling strategy is critical.

First we computed the first four moments of queue length at each queue and we see very accurate results in comparison with simulation in Table 6. All of the model's results lie well inside the confidence bands and, at queue 1, the percentage error ranges from about 0.05% for the mean to about 2.5% on the fourth moment. The accuracy at queue 2 is even better.

We note the low standard deviations, 2.909 and 2.828 (calculated from the moments), expected from JSQ scheduling, and also computed the correlation between the queues to be high at 0.794 (0.805 ± 0.009 by simulation), again as we would expect for JSQ.

Using moments calculated according to the procedure described in section 4.3.4, we created Table 7, comparing the first four moments of response time so obtained with the corresponding simulation. Once again there was a very close match, all model-results lying inside the 95% confidence bands and numerical errors ranging from about 1-4%.

Finally, the probability density function of response time was computed by inverting its Laplace transform numerically. As before, we then compared this with the corresponding regenerative simulation. The results are shown in Fig. 8 and show excellent agreement once more.

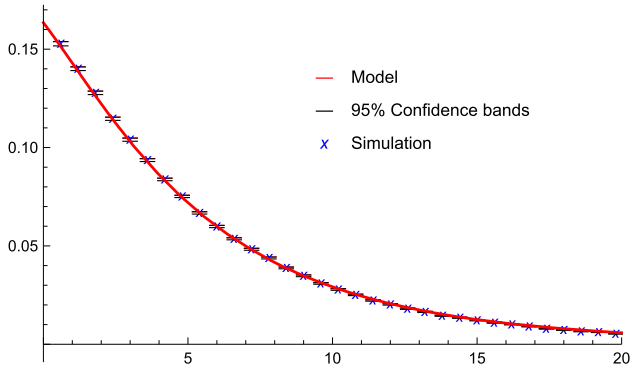


Fig. 8. Response time density ($w(t)$) for two JSQ servers with parameters $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$: comparison with simulation.

4.3.6. A static scheduling model

It is interesting to compare this JSQ model with other, static scheduling policies for the same two servers with total arrival rate λ. The obvious policy would be to split the arrival stream so that each queue gets arrivals at rate proportional to its service rate; i.e. arrival rates $p_1\lambda$ and $p_2\lambda$ to queue 1 and queue 2, respectively, where $p_1 = \mu_1/(\mu_1+\mu_2)$ and $p_2 = \mu_2/(\mu_1+\mu_2)$. The two queues then have exponential response time densities with parameters $\mu_1 - p_1\lambda$ and $\mu_2 - p_2\lambda$. Notice that not any choice of probabilities p_1, p_2 give equilibrium, which requires $p_1\lambda < \mu_1$ and $p_2\lambda < \mu_2$; i.e. $1 - \mu_2/\lambda < p_1 < \mu_1/\lambda$. Hence in our case we require $0.2 < p_1 < 0.4$. The load balanced choice above has $p_1 = \mu_1/(\mu_1+\mu_2) = 1/3,\ p_2 = 2/3$ and yields equal utilizations 0.8333 at the two queues.

The response time density in the static scheduling model has Laplace transform

$$p_1\frac{\mu_1 - p_1\lambda}{s+\mu_1-p_1\lambda} + p_2\frac{\mu_2 - p_2\lambda}{s+\mu_2-p_2\lambda}$$

which is easily differentiated to give the first four moments as 10, 225, 8 437.5 and 455 625, very much higher than those achieved by JSQ scheduling.

5. Conclusion

We have proposed a unified method for the solution of linear bivariate functional equations using results from the theory of functions of complex variables, mapping the equation at a discrete set of points onto a set of linear algebraic equations, solving these and then interpolating to find an approximate solution to the functional equation on a closed contour. The solution at any point inside the contour can then be obtained by contour integration of the interpolation, using Cauchy's integral formula. Functional equations of this kind are important in stochastic modelling because there is a variety of problems involving quite simple systems that have not been solved numerically and new ones are arising. This paper has considered some of these and has shown that accurate numerical solutions for moments and probability density functions are possible, computing detailed numerical solutions for three diverse problems. Accuracy has been found to be excellent in comparison to the few known results available and simulation with tight confidence bands.

Of particular interest is the first numerical solution to the 70-year old join-shortest-queue problem. This included a new technique for handling partial generating functions, which are the key to this and several problems in scheduling, load balancing and other areas. It has thereby become possible to find queue length moments as well as response time moments and densities when the queueing discipline is FCFS to high accuracy. In practice, JSQ policies are usually adopted in combination with PS queueing discipline, and this is a viable area for future work. One reason for confidence is that once the shortest queue has been joined by a tagged customer, the discipline in the other queue could equally well be FCFS as far as that customer is concerned. Therefore in the analysis of this system, only one partial derivative would be involved, corresponding to the PS-queue joined. As we saw in section 4.2, our method is able to deal with this efficiently.

To summarize, there are four main "take-aways" in this paper:

1. The use of the matrices M_{d_1,d_2} to deal with derivatives in functional equations (Corollary 1 in Section 2.4.1 and its use in Section 4.2.1);
2. The evaluation of partial generating functions via contour integrals of the full generating function, together with the definition of their representation matrices $P^{\leq}$ and $P^{\geq}$ (Lemma 2 and Proposition 3 in Section 3.2);
3. An algorithm (Algorithm 1) that automates the numerical solution of any linear functional equation via a mapping to a set of linear equations using the matrices specified in Table 2;
4. Accurate numerical solutions to three previously unsolved problems of significance and practical value, in particular JSQ.

6. Appendices

6.1. Mathematical background

Here we summarize the mathematical results used in Sections 2 and 3, referring to [18,14] as sources.

Definition 5. A complex-valued function f of one or more complex variables that is complex differentiable in a neighbourhood of each point in a connected open set $H \subseteq \mathbb{C}$ is said to be *holomorphic* in H.

It is said to be holomorphic at a point $a \in \mathbb{C}$ if there exists $r > 0$ such that f is defined and analytic in $D(a;r)$, where $D(a;r) := \{z \in \mathbb{C} : |z-a| < r\}$.

Note that a holomorphic function is synonymously called *analytic*,[10] the two properties being equivalent for complex functions. We use this term throughout the paper.

Theorem 4 *(Cauchy's integral theorem). If $f(z)$ is analytic everywhere inside the contour (i.e. simple closed curve) C, then*

$$\int_C dz f(z) = 0.$$

Theorem 5 *(Cauchy's integral formula). For a function $f(z)$ that is analytic in a region containing a contour C and its interior, for any z_0 inside C,*

$$f(z_0) = \frac{1}{2\pi i} \int_C \frac{f(z)}{z - z_0} dz, \tag{20}$$

where we assume C is oriented counterclockwise.

Theorem 6 *(Cauchy's differentiation theorem). For a function $f(z)$ that is analytic in a region containing a contour C and its interior, for any z_0 inside C,*

$$f^{(n)}(z_0) = \frac{n!}{2\pi i} \int_C \frac{f(z)}{(z - z_0)^{n+1}} dz, \qquad n = 0, 1, 2...$$

6.2. Polydisks

Definition 6. A set $D \subset \mathbb{C}^n$ is called a polydisk if there are disks $D_1, ..., D_n$ in $\mathbb{C}$ such that

$$D = \prod_{j=1}^{n} D_j = \{\vec{z} \mid z_j \in D_j, j = 1, ..., n\}.$$

The set $\prod_{j=1}^{n} \partial D_j$ is called the *distinguished boundary* of D and is denoted by $\partial_0 D$.

Theorem 7. *Let D be an open polydisk in C^n and let $u(\vec{z})$ be a continuous function in $\bar{D}$ (the closure of D), which, in D, is analytic in each z_j when the other z_i $(i \neq j)$ are kept fixed $(1 \leq j \leq n)$. Then*

$$u(\vec{z}) = (2\pi i)^{-n} \int_{\partial_0 D} u(\zeta_1, ...\zeta_n)(\zeta_1 - z_1)^{-1} ...(\zeta_n - z_n)^{-1} d\zeta_1 ... d\zeta_n, \tag{21}$$

where $\partial_0 D$ is the distinguished boundary of D. Hence $u \in \mathbb{C}^\infty(D)$ and is analytic in D, i.e. has n-variable Taylor series. Theorem 6 extends similarly to polydisks, giving arbitrary derivatives.

6.3. Comparison of the Laplace transform with that calculated by the BVP method for a tandem pair of queues with negative customers

The comparison is shown in Fig. 9, where the continuous graph (in blue) is computed by our method of Algorithm 1 and the discrete, red, points are taken from [12].

[10] A function $f(z)$ is analytic at a point p if there exists a neighbourhood of p in which f is equal to a convergent power series.

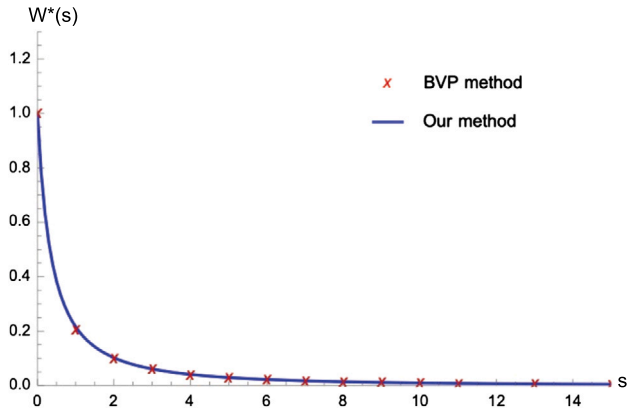


Fig. 9. Laplace Transform $w^*(s)$ of the sojourn time density of a customer not killed for the two-server tandem network where the two queues are FCFS-RCE M/M/1 $\pm$-queues with parameters $\lambda_1^+ = 3$, $\lambda_1^- = 2$, $\lambda_2^+ = 0$, $\lambda_2^- = 2$, $\mu_1 = 2$, $\mu_2 = 1$.

6.4. Proof of Proposition 5

Proposition 5. *The Laplace transform of response time in the pair of M/M/1 queues with JSQ scheduling is*

$$W^*(s) = a_2\left(\frac{\mu_1}{s+\mu_1}\right)[G(\mu_1/(s+\mu_1),1) - G^{\geq}(\mu_1/(s+\mu_1),1)] + a_1\left(\frac{\mu_1}{s+\mu_1}\right)G^{\leq}(\mu_1/(s+\mu_1),1)$$
$$+ a_1\left(\frac{\mu_2}{s+\mu_2}\right)[G(1,\mu_2/(s+\mu_2)) - G^{\leq}(1,\mu_2/(s+\mu_2))] + a_2\left(\frac{\mu_2}{s+\mu_2}\right)G^{\geq}(1,\mu_2/(s+\mu_2))$$

Proof. The Laplace transform of the unconditional response time of a Poisson arrival is, by the Random Observer Property,

$$W^*(s) = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\pi_{ij}\left[(q_1(i,j)w_1^*(s|i) + q_2(i,j)w_2^*(s|j)\right]$$

where $q_k(i,j)$ is the probability that an arrival goes to queue k when the state is (i,j) and $w_k^*(s|n) = \left(\frac{\mu_k}{s+\mu_k}\right)^{n+1}$ is the Laplace transform of the sojourn time density of a customer who joins FCFS queue i when there are n customers already there on arrival $(k = 1, 2)$. Therefore

$$W^*(s) = \sum_{i<j}\pi_{ij}\left(\frac{\mu_1}{s+\mu_1}\right)^{i+1} + \sum_{i>j}\pi_{ij}\left(\frac{\mu_2}{s+\mu_2}\right)^{j+1} + \sum_{i=j}\pi_{ii}\left[a_1\left(\frac{\mu_1}{s+\mu_1}\right)^{i+1} + a_2\left(\frac{\mu_2}{s+\mu_2}\right)^{i+1}\right]$$
$$= \left(\frac{\mu_1}{s+\mu_1}\right)[G(\mu_1/(s+\mu_1),1) - G^{\geq}(\mu_1/(s+\mu_1),1)]$$
$$+ \left(\frac{\mu_2}{s+\mu_2}\right)[G(1,\mu_2/(s+\mu_2)) - G^{\geq}(1,\mu_2/(s+\mu_2))]$$
$$+ a_1\left(\frac{\mu_1}{s+\mu_1}\right)[G^{\leq}(\mu_1/(s+\mu_1),1) + G^{\geq}(\mu_1/(s+\mu_1),1) - G(\mu_1/(s+\mu_1),1)]$$
$$+ a_2\left(\frac{\mu_2}{s+\mu_2}\right)[G^{\leq}(1,\mu_2/(s+\mu_2)) + G^{\geq}(1,\mu_2/(s+\mu_2)) - G(1,\mu_2/(s+\mu_2))]$$

The result then follows after some simplification. □

6.5. Accuracy and cost

6.5.1. Accuracy

Accuracy in the computation of the matrices in Table 2 was considered in the results determining each one in Section 3. This showed errors of $\Theta(n^{-2})$ in the case that there are no derivatives in the functional equation and $\Theta(n^{-1})$ if there are. However, computation of the required generating function requires inversion of a linear combination of these matrices, which matrix will have the same asymptotic error. Since the condition number of the matrix is not large – typically much less than $\Theta(n)$ according to our numerical calculations, the inverse matrix has a similar asymptotic error to those in Table 2. Finally, evaluation of the generating function is the dot product of the inverted matrix and a vector that is known exactly. Hence the asymptotic, worst-case error remains the same, i.e. $\Theta(n^{-2})$ or $\Theta(n^{-1})$, depending on whether there are derivatives in the generating function equation being solved.

However, since each component in this vector is a sum of n real numbers with error $\pm\Theta(n^{-2})$ or $\pm\Theta(n^{-1})$, we expect a final error of $\pm\Theta(n^{-\frac{3}{2}})$ or $\pm\Theta(n^{-\frac{1}{2}})$, for the cases where there are respectively no derivatives or first derivatives.[11]

The numerical results are broadly in line with these estimates, actually better by a factor of 10–100. This is likely because the error-calculations in Sections 2 and 3 are worst-case for every component in the matrices. The accuracy in practice can be expected to be significantly better, as we observe.

6.5.2. Cost

The main computational cost arises from:

1. Construction of the matrices in Table 2 and the calculation of the matrix A from them to form the equation $A\vec{g} = \vec{b}$ in step (3) of Algorithm 1 – or, equivalently, the matrix used to form the analogous equation for the vector $\vec{h}$. All these matrices have dimension $n^2 \times n^2$ (or, more generally, $mn \times mn$) and so contain n^4 elements which are stored and computed element-wise. Thus, both the storage requirement and computational cost to construct these matrices is $\Theta(n^4)$ – in the region of 10^8. The biggest cost is in the computation of the matrices in Table 2, since each element involves a numerical integration. However, once calculated these matrices can be stored for future use in any similar problem with the same value of n. The matrix A is just a linear combination of these and so its computation is quicker. However, it should be noted that in density function calculations, each point in the density function graph – say d of them – requires the Laplace transform at around 16–32 points, hence needing up to $32d$ calculations of the Laplace transform. This is quite a large number, considerably more than n in our applications, so the cost exceeds $\Theta(n^5)$, just to construct the matrices $A(s)$ for each Laplace-point s.
2. The equations for the vector representation of the required generating function now need to be solved by inversion of the matrix A (or matrices $A(s)$). Using commercial mathematical software, this has cost $O(N^3)$, where N is the dimension of the matrix, here $O(n^2)$, giving cost $O(n^6)$. This is the dominant computational cost, but in practice we found the actual, measured cost to be comparable with the calculation of the matrices A. This is probably because, on the one hand, the inversion algorithm used was more efficient than $\theta(n^3)$ and, on the other, the calculation of A requires several matrices each with computational cost $\theta(n^5)$. Our implementation used Mathematica[12] 12.3.1 on an Apple Mac Studio computer with an Apple M1 Max processor chip having 10 cores. This allowed for a degree of parallel processing, which would also improve execution times. For $n = 64$, the costs were moderate – around 30–45 minutes – but for $n = 125$, the greatest value we used, the cost went up to about a day! However, we did not try any optimizations in the numerics, which would reduce the cost many-fold. Moreover, the corresponding simulations ran for several days using the same numerical package.

The remaining computations – of the moments and probability density function – from the inverse matrices have relatively insignificant cost in comparison to the above two cost components.

Data availability

Data will be made available on request.

References

[1] J. Abate, W. Whitt, Numerical inversion of Laplace transforms of probability distributions, ORSA J. Comput. 7 (1) (1995) 36–43.
[2] U. Ayesta, R. Núñez-Queija, M. Verloop, Heavy-traffic analysis of a multiple-phase network with discriminatory processor sharing, Oper. Res. 59 (2011) 648–660.
[3] Chiahon Chien, Batch size selection for the batch means method, in: J.D. Tew, S. Manivannan, D.A. Sadowski, A.F. Seila (Eds.), Proceedings of the 1994 Winter Simulation Conference, 1995, pp. 345–352.
[4] E.G. Coffman Jnr, G. Fayolle, I. Mitrani, Sojourn times in a tandem queue with overtaking: reduction to a boundary value problem, Commun. Stat., Stoch. Models 2 (1986) 43–65.
[5] G. Fayolle, P. Flajolet, M. Hofri, On a functional equation arising in the analysis of a protocol for a multi-access broadcast channel, Adv. Appl. Probab. 18 (1986) 441–472.
[6] Guy Fayolle, R. Iasnogorodski, Isi Mitrani, The distribution of sojourn times in a queueing network with overtaking: reduction to a boundary problem, in: Ashok K. Agrawala, Satish K. Tripathi (Eds.), Performance '83, Proceedings of the 9th International Symposium on Computer Performance Modelling, Measurement and Evaluation, College Park, Maryland, USA, May 25-27, 1983, North-Holland, 1983, pp. 477–486.
[7] L. Flatto, H.P. McKean, Two queues in parallel, Commun. Pure Appl. Math. 30 (2) (1977) 255–263, https://doi.org/10.1002/cpa.3160300206.
[8] E. Gelenbe, Random neural networks with positive and negative signals and product form solution, Neural Comput. 1 (4) (1989) 502–510.
[9] E. Gelenbe, G-networks with triggered customer movement, J. Appl. Probab. 30 (1993) 742–748.
[10] Peter W. Glynn, Simulation algorithms for regenerative processes, in: Shane G. Henderson, Barry L. Nelson (Eds.), Simulation, in: Handbooks in Operations Research and Management Science, vol. 13, Elsevier, 2006, pp. 477–500.
[11] P.G. Harrison, E. Pitel, Sojourn times in single-server queues with negative costumers, J. Appl. Probab. 30 (1993) 943–963.
[12] P.G. Harrison, E. Pitel, Response time distributions in tandem G-networks, J. Appl. Probab. 32 (1995) 224–246.
[13] P.G. Harrison, N.M. Patel, Performance Modelling of Communication Networks and Computer Architectures, Addison-Wesley Longman Ltd., 1993.
[14] L. Hormander, An Introduction to Complex Analysis in Several Variables, Elsevier, 1973.
[15] Wolfram Research, Inc., Mathematica, version 14.0, https://www.wolfram.com/mathematica, 2024, Champaign, IL.

[11] Intuitively, this is because there will generally be some cancellation amongst the errors. A multiple of $\Theta(n^{\frac{1}{2}})$ comes from the Central Limit Theorem when summing the n real numbers, and this is multiplied by the individual errors $\pm\Theta(n^{-2})$ or $\pm\Theta(n^{-1})$, respectively.

[12] See [15].

[16] J.F.C. Kingman, Two similar queues in parallel, Ann. Math. Stat. 32 (4) (1961) 1314–1323, https://doi.org/10.1214/aoms/1177704869.
[17] D. Mitra, J.A. Morrison, Asymptotic expansions of moments of the waiting time in a shared processor of an interactive system, Adv. Appl. Probab. 15 (1983) 813–839.
[18] H.A. Priestley, Introduction to Complex Analysis, second edition (2nd ed.), Oxford University Press, 2003.
[19] M. Taube-Netto, Two queues in tandem attended by a single server, Oper. Res. 25 (1997) 140–147.
[20] J. Walrand, P. Varaiya, Sojourn times and the overtaking condition in Jacksonian networks, Adv. Appl. Probab. 12 (4) (1980) 1000–1018, https://doi.org/10.2307/1426753.