On Computation of Approximate Solutions to Large-Scale Backstepping Kernel Equations via Continuum Approximation*

Jukka-Pekka Humaloja[a], Nikolaos Bekiaris-Liberis[a]

[a]*Department of Electrical and Computer Engineering, Technical University of Crete, University Campus, Akrotiri, Chania, 73100, Greece*

Abstract

We provide two methods for computation of continuum backstepping kernels that arise in control of continua (ensembles) of linear hyperbolic PDEs and which can approximate backstepping kernels arising in control of a large-scale, PDE system counterpart (with computational complexity that does not grow with the number of state components of the large-scale system). In the first method, we provide explicit formulae for the solution to the continuum kernels PDEs, employing a (triple) power series representation of the continuum kernel and establishing its convergence properties. In this case, we also provide means for reducing computational complexity by properly truncating the power series (in the powers of the ensemble variable). In the second method, we identify a class of systems for which the solution to the continuum (and hence, also an approximate solution to the respective large-scale) kernel equations can be constructed in closed form. We also present numerical examples to illustrate computational efficiency/accuracy of the approaches, as well as to validate the stabilization properties of the approximate control kernels, constructed based on the continuum.

Keywords: Backstepping kernels computation, Large-scale hyperbolic systems, PDE continua

1. Introduction

Exponentially stabilizing feedback control laws based on the backstepping method [1] have been developed for several classes of hyperbolic PDEs in recent years, see [2–10]. The application of the method requires computing the backstepping control (and observer) gains, which involves solving the respective kernel PDEs. In certain instances, as in the case of 2×2 linear, hyperbolic systems with constant coefficients [11], it is possible to derive the solution to the kernel PDEs in closed-form; whereas, in general, the solution to the kernel PDEs has to be computed (i.e., approximated) numerically [12–16]. However, for large-scale hyperbolic systems, such numerical schemes for solving the kernel PDEs exhibit computational complexity that grows with the number of state components of the PDEs [17, 18]. Motivated by this, in the present paper, we provide tractable computational tools for constructing exponentially stabilizing feedback laws (by solving the respective kernel PDEs) for large-scale [3] and continua [2] of hyperbolic PDEs. We achieve this in two ways; by adapting the power series method [13] to solve continuum kernel PDEs and by providing closed-form (exact) solutions for a class of such kernel PDEs. As we have shown in [17, 18], the continuum control kernel provides stabilizing feedback control gains for the respective large-scale hyperbolic PDE system as well.

For large-scale, 1-D, linear hyperbolic systems, one of the most efficient approaches to compute backstepping control kernels is to utilize a closed-form, explicit solution. However, an exact, closed-form solution is available only for specific classes of hyperbolic systems, such as, for 2×2 systems with constant coefficients [11]. An alternative in providing explicit, backstepping control kernels, constitutes in computing approximate (yet explicit) solutions to the respective kernels PDEs via, for example, late lumping [14], neural operators [16], and power series [13, 19–21] -based approaches. In particular, the power series method provides a simple and flexible approach for computation of approximate backstepping kernels, thus making it a suitable candidate for computation of the solutions to continua and large-scale kernels PDEs.

As the main contribution of the paper, we present a triple power series-based approach to compute the solution to continuum kernel equations on a prismatic 3-D domain, thus, providing a systematic, computational tool for constructing stabilizing feedback laws for continua of hyperbolic PDEs. In turn, utilizing our recent results on continuum approximation of large-scale systems of hyperbolic PDEs [17, 18], the stabilizing feedback laws obtained for the continuum can be directly employed for stabilizing the corresponding large-scale system. We provide theoretical guarantees that the power series approximation is convergent, provided that the parameters of the continuum kernel PDEs are analytic, and present an algo-

*Funded by the European Union (ERC, C-NORA, 101088147). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

rithm to compute the solution. Moreover, we propose an order-reduction method for the power series approach, which can potentially reduce the computational complexity of the approach from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2)$, where N is the order of the power series approximation. In either case, we show that the power series method to compute continuum kernels results in providing stabilizing control kernels for the respective large-scale system with computational complexity that does not grow with the number of PDE state components (in contrast to the case of employing the power series approach to solve the large-scale kernel equations). We then apply the proposed approach to a stabilization problem of a large-scale hyperbolic system of PDEs, where an exponentially stabilizing state-feedback gain can be computed efficiently by combining the continuum approximation approach [17, 18] and the proposed reduced-order, triple power series method. We also conduct numerical experiments to illustrate the computational effectiveness/accuracy of the (triple) power series method.

We then identify special cases, in which the solution to the continuum kernel equations can be expressed in closed form. This may be viewed as reminiscent of [11], where explicit kernels are computed for stabilization of 2×2 linear hyperbolic PDE systems with constant parameters. This special class of continuum kernel equations is characterized by specific conditions on the respective continuum parameters, under which we provide a closed-form solution in a constructive manner. Because the conditions imposed concern the continuum parameters, there is some flexibility degree for their satisfaction in the case in which the continuum parameters involved are obtained as continuum approximation (which may not be unique) of the respective sequences of parameters of the large-scale system counterpart (see [17, 18]). We also present a numerical example for which these conditions on the continuum kernel equations are satisfied, while a closed-form solution cannot be derived for the respective large-scale system. Thus, such closed-form solution provides explicit stabilizing control gains for the continuum, as well as for any large-scale system of hyperbolic PDEs that can be approximated by such a continuum, even when a closed-form solution for the original, large-scale kernel equations may not be available (see also [17, 18]).

The paper is organized as follows. In Section 2, we present the large-scale kernel equations (associated to a large-scale system of hyperbolic PDEs) and their continuum approximation. In Section 3, we present the power series approach to solving the continuum kernel PDEs, including an explicit algorithm and a potential order-reduction method. In Section 4, we identify sufficient conditions for the parameters of the continuum kernel equations, under which the solution can be expressed in closed form. In Section 5, we demonstrate the accuracy and convergence rate of the power series approximation in a numerical example. In Section 6, we apply the power series method to the stabilization problem of a large-scale system of hyperbolic PDEs, where a closed-form solution to the kernel equations (large-scale or continuum) is not available. Finally, Section 7 contains concluding remarks.

2. Large-Scale Kernel Equations and Their Continuum Approximation

In this section, we present the considered class of large-scale hyperbolic kernel equations, and how they can be approximated by respective continua of kernel equations by virtue of our earlier works [17, 18]. This provides the basis for the development of computational tools to approximate the solution of the large-scale kernel equations by solving the respective continua of kernel equations instead.

For $i = 1, \dots, n$, where n is large, consider kernel equations of the form [3, 17]

$$\mu(x)\partial_x k^i(x,\xi) - \lambda_i(\xi)\partial_\xi k^i(x,\xi) = \lambda_i'(\xi)k^i(x,\xi) + \frac{1}{n}\sum_{j=1}^{n}\sigma_{j,i}(\xi)k^j(x,\xi) + \theta_i(\xi)k^{n+1}(x,\xi), \tag{1a}$$

$$\mu(x)\partial_x k^{n+1}(x,\xi) + \mu(\xi)\partial_\xi k^{n+1}(x,\xi) = -\mu'(\xi)k^{n+1}(x,\xi) + \frac{1}{n}\sum_{j=1}^{n}W_j(\xi)k^j(x,\xi), \tag{1b}$$

where $\mu > 0, \lambda_i > 0, \sigma_{i,j}, \theta_i, W_i$, and q_i are given parameters and $(k^i)_{i=1}^{n+1}$ is the sought solution, i.e., the exact kernels. The equations (1) are defined on the triangular domain

$$\mathcal{T} = \left\{(x,\xi) \in [0,1]^2 : 0 \le \xi \le x \le 1\right\}, \tag{2}$$

and equipped with boundary conditions

$$k^i(x,x) = -\frac{\theta_i(x)}{\lambda_i(x)+\mu(x)}, \tag{3a}$$

$$\mu(0)k^{n+1}(x,0) = \frac{1}{n}\sum_{j=1}^{n}q_j\lambda_j(0)k^j(x,0), \tag{3b}$$

for all $x \in [0,1]$. Our aim is to find approximate solutions to (1), (3) by solving the corresponding continuum kernel equations of the form [2, 17]

$$\mu(x)\partial_x k(x,\xi,y) - \lambda(\xi,y)\partial_\xi k(x,\xi,y) - \theta(\xi,y)\bar{k}(x,\xi) = k(x,\xi,y)\partial_\xi\lambda(\xi,y) + \int_0^1 \sigma(\xi,\eta,y)k(x,\xi,\eta)d\eta, \tag{4a}$$

$$\mu(x)\partial_x\bar{k}(x,\xi) + \mu(\xi)\partial_\xi\bar{k}(x,\xi) = -\mu'(\xi)\bar{k}(x,\xi) + \int_0^1 W(\xi,y)k(x,\xi,y)dy, \tag{4b}$$

where $\mu > 0, \lambda > 0, \sigma, \theta, W$, and q are given parameters and $(k, \bar{k})$ is the sought solution, i.e., the continuum kernels. The equations (4) are defined on a prismatic domain

$$\mathcal{P} = \left\{(x, \xi, y) \in [0,1]^3 : (x, \xi) \in \mathcal{T}\right\}, \tag{5}$$

and equipped with boundary conditions

$$k(x, x, y) = -\frac{\theta(x, y)}{\lambda(x, y) + \mu(x)}, \tag{6a}$$

$$\mu(0)\bar{k}(x, 0) = \int_0^1 q(y)\lambda(0, y)k(x, 0, y)dy, \tag{6b}$$

for all $x \in [0, 1]$ and for almost every $y \in [0, 1]$.

The kernel equations (1), (3) appear in backstepping control of $n + 1$ hyperbolic PDEs [3] (see (B.1), (B.2) in Appendix B), where the solution $\left(k^i\right)_{i=1}^{n+1}$ provides an exponentially stabilizing state feedback gain for the system (see (B.3)). More recently [2], the backstepping control methodology has been extended to continua of hyperbolic PDEs (see (B.4), (B.5) in Appendix B), where the exponentially stabilizing state feedback gain is obtained from the solution $(k, \bar{k})$ to the continuum kernel equations (4), (6) (see (B.6)). In our recent work [17, 18], we have shown that the continuum kernel equations (4), (6) can be interpreted as an approximation of (1), (3) when n is sufficiently large (see [17, Lem. 4.2 & Lem. 4.3]). Thus, the exponentially stabilizing state feedback gain for the large-scale $n+1$ hyperbolic PDE can be approximated from the solution to (4), (6), without compromising the exponential stability of the closed-loop system (see [17, Thm 4.1]). The relation between the parameters of (1), (3) and (4), (6) was established in [17, 18] as

$$\lambda(x, i/n) = \lambda_i(x), \tag{7a}$$

$$W(x, i/n) = W_i(x), \tag{7b}$$

$$\theta(x, i/n) = \theta_i(x), \tag{7c}$$

$$\sigma(x, i/n, j/n) = \sigma_{i,j}(x), \tag{7d}$$

$$q(i/n) = q_i, \tag{7e}$$

for all $x \in [0, 1]$ and $i, j = 1, 2, \ldots, n$, where the parameters of (4), (6) are assumed to be continuous in y and η so that the pointwise evaluations are well-defined. Under (7), we have shown in [17, 18] that the solution to (1), (3) converges to the solution of (4), (6) (in the L^2 sense in y; see [17, Lem. 4.3])[1]. Consequently, the solution of (4), (6) can be used to approximate the solution to (1), (3) and to construct an exponentially stabilizing feedback gain for a large-scale $n + 1$ hyperbolic system of PDEs, which is our main motivation of studying solving (4), (6).

[1]This relies on interpreting the continuum parameters of (4), (6) as the limits of sequences of functions defined as piecewise constant in y, on intervals of the form $((i-1)/n, i/n]$ (see [17, Lem. 4.2]).

3. Power Series Approach to Solving Continuum Kernel Equations

In this section, we present a power series-based approach to solving the continuum kernel equations (4), (6) for analytic parameters $\mu, \lambda, \sigma, \theta, W$, and q. We show that, in this case, the solution $(k, \bar{k})$ to (4), (6) is analytic, and propose a numerical procedure to approximate the solution to arbitrary accuracy. Corresponding approaches have been adopted to solve various kernel PDEs [13, 20, 21], but our developments are new for continua of kernel PDEs (4), (6). Moreover, we provide a computational metric to assess the accuracy of the approximate solution, and consider computing reduced-order approximations to reduce the computational complexity of the proposed approach.

3.1. Convergence of Power Series Representation

As per [13], the idea is to find the solution to (4), (6) as a power series, which in case of $k(x, \xi, y)$ is a triple power series

$$k(x, \xi, y) = \sum_{\ell=0}^{\infty}\sum_{i=0}^{\infty}\sum_{j=0}^{i} K_{ij\ell}x^{i-j}\xi^j y^\ell, \tag{8}$$

whereas for $\bar{k}(x, \xi)$ the power series representation is

$$\bar{k}(x, \xi) = \sum_{i=0}^{\infty}\sum_{j=0}^{i} \bar{K}_{ij}x^{i-j}\xi^j. \tag{9}$$

Similarly, the parameters of (4), (6) are represented by the series

$$\lambda(x, y) = \sum_{i=0}^{\infty}\sum_{j=0}^{i} \lambda_{ij}x^{i-j}y^j, \tag{10a}$$

$$\mu(x) = \sum_{i=0}^{\infty} \mu_i x^i, \tag{10b}$$

$$\theta(\xi, y) = \sum_{i=0}^{\infty}\sum_{j=0}^{i} \theta_{ij}\xi^{i-j}y^j, \tag{10c}$$

$$W(\xi, y) = \sum_{i=0}^{\infty}\sum_{j=0}^{i} W_{ij}\xi^{i-j}y^j, \tag{10d}$$

$$\sigma(x, \eta, y) = \sum_{\ell=0}^{\infty}\sum_{i=0}^{\infty}\sum_{j=0}^{i} \sigma_{ij\ell}x^{i-j}\eta^j y^\ell, \tag{10e}$$

$$q(y) = \sum_{i=1}^{\infty} q_i y^i, \tag{10f}$$

where the coefficients are obtained from the Taylor series of the respective parameters. Similarly to [13], we consider the parameters and the kernels appearing in (8)–(10) complex-valued, and by a polydisk we refer to $\mathcal{D}_L \times \mathcal{D}_L \times \mathcal{D}_L$ (or $\mathcal{D}_L \times \mathcal{D}_L$ if only two spatial variables are involved), where $\mathcal{D}_L$ is a complex-valued open disk centered at the origin of radius L, i.e.,

$$\mathcal{D}_L = \{z \in \mathbb{C} : |z| < L\}. \tag{11}$$

For the power series in (8)–(10) to converge in the domain of definition of the kernel equations (4), the involved functions have to be analytic on polydisks with radius larger than one.[2]

By [2, Thm 3], under sufficient continuity assumptions on the parameters of (4), (6), there exists a unique solution $(k, \bar{k})$ to (4), (6). However, representing the solution as a power series requires the solution to be analytic, so that stronger assumptions have to be imposed on the parameters of (4), (6). On the other hand, if an analytic solution exist, it is uniquely given by the power series (8), (9), because of the uniqueness of the solution and the Taylor series representation. Similar to the results of [13], we utilize the well-posedness result [2, Thm 3] of (4), (6) to state the following.

Theorem 1. *If the parameters of* (4), (6) *are analytic on polydisks with radii larger than one, so that they can be represented as the series in* (10)*, and* $|\lambda(x,y)| > 0$ *for all* $x, y \in [0,1]$*,* $|\mu(x)| > 0$ *for all* $x \in [0,1]$*, then the series defined in* (8), (9) *converge. That is, they define analytic functions on polydisks with radii larger than one, which are the unique solution to the kernel equations* (4), (6)*.*

Proof. The proof follows the same steps as the argument for the analogous result for the 1+1 system in [13, Thm 3], that is, by complexifying the kernel well-posedness proof of [2, Thm 3]. In more detail, extend the successive approximation series given in [2, (109), (110)] to polydisks, which requires considering the integrals along the characteristic curves as line integrals in complex spaces.[3] As all the parameters are assumed to be analytic, the products and inner products appearing inside the integrals are analytic as well, and such integrals of analytic functions are independent of the integration path. Consequently, it follows by recursion that each term in the successive approximation series is analytic, being composed of integrals and (inner) products of analytic functions. The uniform convergence of the complexified series of successive approximations follows by the Weierstrass M-test (see, e.g., [22, Thm 7.10]) after derivation of similar estimates as in [2, (111)–(116), (121)–(128)] for the complexified parameters. □

3.2. Computation of Approximate Solution with Truncated Power Series

Based on Theorem 1, we can construct arbitrarily accurate approximations of the solution to (4), (6), assuming that the involved parameters are analytic, by truncating the infinite series (8)–(10) up to some sufficiently high order N. This results in finding an approximate solution $(k^{\mathrm{a}}, \bar{k}^{\mathrm{a}})$ to (4), (6) as

$$k^{\mathrm{a}}(x,\xi,y) = \sum_{\ell=0}^{N} \sum_{i=0}^{N-\ell} \sum_{j=0}^{i} K_{ij\ell}^{\mathrm{a}} x^{i-j} \xi^{j} y^{\ell}, \tag{12a}$$

$$\bar{k}^{\mathrm{a}}(x,\xi) = \sum_{i=0}^{N} \sum_{j=0}^{i} \bar{K}_{ij}^{\mathrm{a}} x^{i-j} \xi^{j}, \tag{12b}$$

for approximate parameters

$$\lambda^{\mathrm{a}}(x,y) = \sum_{i=0}^{N} \sum_{j=0}^{i} \lambda_{ij} x^{i-j} y^{j}, \tag{13a}$$

$$\mu^{\mathrm{a}}(x) = \sum_{i=0}^{N} \mu_i x^{i}, \tag{13b}$$

$$\theta^{\mathrm{a}}(\xi,y) = \sum_{i=0}^{N} \sum_{j=0}^{i} \theta_{ij} \xi^{i-j} y^{j}, \tag{13c}$$

$$W^{\mathrm{a}}(\xi,y) = \sum_{i=0}^{N} \sum_{j=0}^{i} W_{ij} \xi^{i-j} y^{j}, \tag{13d}$$

$$\sigma^{\mathrm{a}}(x,\eta,y) = \sum_{\ell=0}^{N} \sum_{i=0}^{N-\ell} \sum_{j=0}^{i} \sigma_{ij\ell} x^{i-j} \eta^{j} y^{\ell}, \tag{13e}$$

$$q^{\mathrm{a}}(y) = \sum_{i=1}^{N} q_i y^{i}. \tag{13f}$$

Note that while the parameters are approximated by truncating the series (10), the coefficients of (12) may not (for any finite N) match the corresponding coefficients of (8), (9) (which are unknown). A numerical procedure to compute the coefficients of the truncated power series approximation (12) by least-squares is presented in Algorithm 1, and the asymptotic convergence of the proposed approximation to the exact solution (8), (9) is established in Proposition 1.[4]

Proposition 1. *Under the conditions of Theorem 1, the approximate solution* (12) *produced by Algorithm 1 converges asymptotically to the exact solution* (8), (9) *as* $N \to \infty$*.*

Proof. For any N, refer to the linear system of equations of Algorithm 1 as $\mathbf{A}_N \mathbf{x}_N = \mathbf{b}_N$, and denote the coefficients of the (exact) truncated power series (8), (9) of order N by $\mathbf{x}_N^{\mathrm{e}}$. By definition, a least-squares solution to $\mathbf{A}_N \mathbf{x}_N = \mathbf{b}_N$, denoted by $\mathbf{x}_N^{\mathrm{a}}$, minimizes $\|\mathbf{A}_N \mathbf{x}_N - \mathbf{b}_N\|_2$. Thus, for all N, we have

$$\|\mathbf{A}_N \mathbf{x}_N^{\mathrm{a}} - \mathbf{b}_N\|_2 \leq \|\mathbf{A}_N \mathbf{x}_N^{\mathrm{e}} - \mathbf{b}_N\|_2. \tag{14}$$

[2] Alternatively, the power series can be developed with respect to some other point than the origin, in which case polydisks with smaller radii are sufficient, provided that the prism $\mathcal{P}$ (or the triangle $\mathcal{T}$ in case of $\bar{k}$) is covered.

[3] The complexified equations are identical to [2, (128), (129)]; only the interpretation (real vs. complex) changes.

[4] As presented here, the algorithm can be implemented, e.g., in MATLAB by utilizing symbolic computations, and such an implementation is available at `https://github.com/jphumaloja/Continuum-Kernels-Power-Series/`. Another approach would be to adapt the double series vector-matrix framework from [21, Sect. II.B], which is computationally more efficient, but potentially also more laborious to implement for the triple power series appearing in our computations.

Data: Parameters $\lambda, \mu, \sigma, \theta, W, q$ of (4), (6).
Result: Truncated power series approximation (12) for $(k, \bar{k})$.
Initialization:
- Choose approximation order N.
- Construct the truncated series (12), (13) (with unknowns the coefficients of (12)).
- Insert the truncated series into (4), (6).
- Store the coefficients of each term $x^i \xi^{j-i} y^\ell$ from (4), (6) into a list.
- Initialize matrix $\mathbf{A}$ and vector $\mathbf{b}$ based on the number of unknown coefficients $K^{\mathrm{a}}_{ij\ell}, \bar{K}^{\mathrm{a}}_{ij}$ and the length of the list.

while *not at the end of the list* **do**
- Check next element in the list of coefficients.
- **if** *coefficient has* $K^{\mathrm{a}}_{ij\ell}$ *or* $\bar{K}^{\mathrm{a}}_{ij}$ **then**
  - Insert the numerical value to the corresponding position in $\mathbf{A}$.
- **else**
  - Insert the numerical value to the corresponding position in $\mathbf{b}$.
- **end**

end
Solve for the unknown coefficients from $\mathbf{A}\mathbf{x}^{\mathrm{a}} = \mathbf{b}$ by least squares.
Insert the obtained values $\mathbf{x}^{\mathrm{a}}$ to (12).

Algorithm 1: Computation of the power series approximation (12).

In the limit case $N \to \infty$, by the uniqueness of the solution to (4), (6) and the uniqueness of the power series representation, the coefficients $\mathbf{x}^{\mathrm{e}}_\infty$ of (8), (9) uniquely solve the (infinite) set of linear equations $\mathbf{A}_\infty \mathbf{x}_\infty = \mathbf{b}_\infty$, i.e., $\|\mathbf{A}_\infty \mathbf{x}^{\mathrm{e}}_\infty - \mathbf{b}_\infty\|_2 = 0$. Combining this with (14) gives $\mathbf{x}^{\mathrm{a}}_\infty = \mathbf{x}^{\mathrm{e}}_\infty$. Thus, the coefficients of (12) tend to the coefficients of (8), (9) as $N \to \infty$, and the claim follows. □

As stated in Algorithm 1, the solution to the kernel equations is obtained by replacing the parameters and the solution in the kernel equations (4), (6) by the power series approximations (12), (13). In turn, solving for the unknown coefficients $K^{\mathrm{a}}_{ij\ell}$ and $\bar{K}^{\mathrm{a}}_{ij}$ such that the equations are satisfied, matching the coefficients of powers of x, ξ, and y appearing in the series. In particular, as also noted in [13, Sect. III.D], it is reasonable to express the boundary condition (6a) as

$$(\lambda(x,y) + \mu(x))k(x,x,y) = -\theta(x,y), \tag{15}$$

so that both sides are polynomials in x and y, because the original form (6a) is not directly compatible with the power series approach.[5] Finally, we note that the set of linear equations considered in Algorithm 1 is usually overdetermined. This is due to the parameters and solution being approximated by power series of order N, so that any multiplication in (4), (6) results in the appearance of terms up to order $2N$, which contribute to the equations that have to be satisfied by the series coefficients.

By virtue of Proposition 1, the accuracy of the approximate solution obtained with Algorithm 1 can be assessed based on the residual $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$ of the least-squares fit. If the residual is large, then one should increase the approximation order N. Finally, as the convergence of power series is uniform (in space), the obtained approximate solution is arbitrarily close to the exact solution pointwise, meaning that we also get arbitrarily accurate approximations for the state feedback gains $k(1,\xi,y)$ and $\bar{k}(1,\xi)$ employed in the backstepping control law (see (B.6)). The spatial error of the approximation can in fact be estimated, similarly to the remainder of the truncated series (8), (9), by expressions of the form (see [23, Prop. 1.2.2])

$$|R_N(x,\xi,y)| \leq M \frac{L^{D(N+1)}}{(N+1)!}, \tag{16}$$

where $D = 2, 3$ for (9), (8), respectively, and M is a bound depending on the derivatives of order $N+1$ of the approximated function.

The bound M can (technically) be estimated by deriving the kernel equations for the derivatives of order $N+1$ of $(k, \bar{k})$ by differentiating (4), (6) with respect to (x, ξ, y) (see, e.g., [7, App. A.4] for the case of 2×2 kernels). The resulting kernel equations are essentially of the same form as (4), (6) (with the lower-order derivatives of $(k, \bar{k})$ acting as additional parameters), and thus, their solution can be upper-bounded using the method of successive approximations as in [2, Sect. VI]. However, in order to compute such a bound, all derivatives of $(k, \bar{k})$ up to order $N+1$ would need to be estimated, implying that $\frac{(N+2)(N+3)}{2}$ estimates for the derivatives of $\bar{k}$ and $\frac{(N+1)(N+2)(2N+12)}{12} + N + 2$ estimates for the derivatives of k would have to be computed, which has the same computational complexity $\mathcal{O}(N^3)$ as the proposed power series procedure for actually computing the kernels themselves. (see Section 3.3). In other words, it would be more practically useful to evaluate the truncation error from the least-squares error of Algorithm 1, rather than trying to quantify it through the power series residual.

[5] Alternatively, one could compute a power series approximation separately for the right-hand side of (6a). However, one should note that the analyticity of λ, μ, and θ does not generally imply that the fraction on the right-hand side of (6a) would be analytic. Hence, the form (15) is preferable, where the functions are replaced by their power series approximation when solving for the unknown coefficients.

3.3. Computational Complexity of the Power Series-Based Kernels Approximation

The number of unknown coefficients in the truncated power series (12) is $\frac{(N+1)(N+2)}{2}$ for $\bar{K}^{\mathrm{a}}_{ij}$ and

$$\sum_{i=0}^{N} \frac{(i+1)(i+2)}{2} = \frac{N(N+1)(2N+10)}{12} + N + 1, \quad (17)$$

for $K^{\mathrm{a}}_{ij\ell}$, meaning that we have that many unknowns to solve for, whose number increases proportionally to N^3. Thus, the computational complexity of the triple power series approach grows cubically with the approximation order N. This is consistent with quadratic growth reported in [13] for the double power series.

If we view (4), (6) as an approximation of (1), (3) and think of solving (1), (3) using a power series approach, this would involve $n+1$ double power series, one for each of the components k^i (similar to $\bar{k}$ in the continuum case), meaning that the power series approach to solve (1), (3) would involve solving for $(n+1)\frac{(N+1)(N+2)}{2}$ unknown coefficients. Thus, the growth in computational complexity to solve (1), (3) based on a power series approach is $\mathcal{O}(nN^2)$, whereas for solving the continuum kernel equations (4), (6) it is $\mathcal{O}(N^3)$. Hence, whenever it is possible to choose (roughly) $N < n$ without compromising the approximation accuracy, the continuum approximation approach of solving (4), (6) reduces the computational complexity of the power series approach compared to solving (1), (3). More importantly, in the case of solving (4), (6), the computational complexity does not scale with the number of components of the $n+1$ system.

In the next subsection, we discuss a potential order reduction method with respect to the ensemble variable y, which may reduce the complexity of the triple power series from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2)$. That is, instead of a generic order for the truncated power series (12), we employ a separate, lower order $N_y < N$ for the powers of y, which leads to a smaller number of unknown coefficients, and hence, to computational complexity of order $\mathcal{O}(N_y N^2)$. Thus, assuming that the order N_y can be kept constant, the computational complexity only grows quadratically with respect to N as opposed to cubically in (12). Moreover, compared to the complexity $\mathcal{O}(nN^2)$ of solving (1), (3) with a power series approach, the reduced-order approach with respect to y further reduces computational complexity whenever (roughly) $N_y < n$. In both cases of computing the power series corresponding to (4), (6), computational complexity does not grow with n, in contrast to the case of computing the power series corresponding to (1), (3). The computational complexities of the three considered approaches are summarized in Table 1 below.

As our continuum approximation approach asks for sufficiently large n, we, in general, expect N or N_y to be much smaller than n, or, more accurately, our approach, theoretically, relies on the assumption that n is (sufficiently) large. Thus, a more fair comparison would be to compare N with N_y so that, in addition to having computational complexity not growing with n, we further reduce computational complexity from N^3 to $N_y N^2$, despite that, from a purely algebraic viewpoint, we have to also compare N_y with n so that to illustrate (algebraically) that complexity $O(nN^2)$ is worse also as compared with $O(N_y N^2)$.

Approach	Full (4), (6)	(1), (3)	R-O y (4), (6)
Complexity	$\mathcal{O}(N^3)$	$\mathcal{O}(nN^2)$	$\mathcal{O}(N_y N^2)$

Table 1: Computational complexities of the power series approaches to solving (1), (3) and (4), (6) with a potentially reduced-order (R-O) in y series.

3.4. On Order Reduction with Respect to y

Instead of having a generic degree N for the power series approximations (12), we can choose a lower degree for some of the spatial variables. Considering that y is the ensemble variable (when interpreting (4), (6) as the continuum of (1), (3)), and there is some freedom involved in constructing the continuum parameters satisfying (7) (see [17]), it may be possible to obtain sufficiently accurate power series approximations with a much lower order in y. One such particular case is when all the parameters are, or can be approximated by, low-order polynomials in y. Based on the power series approximation in (12a), we can construct a reduced-order approximation in y as

$$k^{\mathrm{a}_y}(x,\xi,y) = \sum_{\ell=0}^{N_y} \sum_{i=0}^{N-\ell} \sum_{j=0}^{i} K^{\mathrm{a}_y}_{ij\ell} x^{i-j} \xi^j y^\ell, \quad (18)$$

for some $N_y < N$. As reported in Table 1, the respective computational complexity to solve for the unknown coefficients is $\mathcal{O}(N_y N^2)$.

Assuming that the parameters λ, θ, and σ are, or can be approximated by, low-order polynomials in y, one can try to approximate the continuum kernels with reduced-order N_y in y. The reason as to why a lower order N_y in y, than in the other spatial variables, would still provide an accurate approximation, is that (4), (6) is not a differential equation in y. Hence, an order in y similar to the respective order of the parameters, determining the spatial dependence of (4a) in y, may be enough to approximate the solution to (4), (6) accurately. On the other hand, considering that particularly (15) has to be satisfied, the order of $\lambda(x,y)k(x,x,y)$ has to be at least the same as the order of $\theta(x,y)$ in y. That is, if the orders of $\lambda(x,y)$ and $\theta(x,y)$ in y are N_λ and N_θ, respectively, then necessarily $N_y \geq N_\theta - N_\lambda$, which provides a lower bound for N_y.[6]

[6] We have observed in numerical experiments that, in general, this lower bound may be sufficient particularly when λ is constant in y, i.e., $N_\lambda = 0$. However, for y-varying λ, it may be more difficult to determine, a priori, a sufficient approximation order N_y in y (without compromising the approximation accuracy).

If the continuum kernel equations (4), (6) are treated as an approximation of the large-scale kernel equations (1), (3), a potential additional benefit of the continuum approximation can be gained particularly when there exists continuum parameters that are low-order polynomials in y, such that the relations (7) are satisfied for the corresponding large-scale parameters, which potentially results in $N_y < n$. For example, this is the case if the large-scale parameters are parametrized by low-order polynomials of i/n, i.e., they form sequences of polynomials in i, in which case the continuum parameters satisfying (7) can be constructed by replacing i/n with y in the expressions of the large-scale parameters.[7] We demonstrate this later on in Section 6.

In general, we note that (7) is not strictly necessary for the continuum approximation to be sufficiently accurate (cf. [17, Rem. 4.4]). Essentially, any polynomial that approximates the distribution of the large-scale parameters as an ensemble (in the L^2 sense in y) is sufficient. For finding such polynomial approximations in practice, many computational software have routines for such purposes, such as, e.g., `polyfit` in MATLAB, which can be potentially utilized in constructing such polynomial continuum approximations. We demonstrate this approach as well in Section 6.

4. On Explicit Solution to Continuum Kernel Equations

In this section, we identify sufficient conditions on the parameters of the kernel equations (1), (3), under which the continuum kernel equations (4), (6) can be solved explicitly. The main idea towards this is to look for separable solutions to the continuum kernel equations, the existence of which requires certain assumptions on the continuum parameters, and respectively, on the parameters of the original large-scale kernel equations.[8] In Section 5, we utilize the explicit solutions to benchmark the approximate power series approach presented in Section 3.

Assumption 1. *For the parameters of the kernel equations* (1), (3)*, we assume that* $\mu(x) = \mu$ *and* $\lambda_i(x) = \lambda_i$*, where* μ *and* λ_i*, for all* $i = 1, \dots, n$*, are positive constants. Moreover, the parameters* $W_i(x), \sigma_{i,j}(x), \theta_i(x)$ *are of the form*

$$W_i(x) = w_i W_x(x), \tag{19a}$$
$$\sigma_{i,j}(x) = s_{1,i} s_{2,j} \sigma_x(x), \tag{19b}$$
$$\theta_i(x) = \vartheta_i \theta_x(x), \tag{19c}$$

for some constants $w_i, s_{1,i}, s_{2,j}, \vartheta_i$ *for all* $i, j = 1, \dots, n$.

Under Assumption 1, the parameters of the continuum kernel equations (4), (6) can be constructed to be separable (with λ constant in x), i.e., there exist functions $\lambda_y, W_y, \theta_y, \sigma_y, \sigma_\eta$ such that

$$\lambda(x, y) = \lambda_y(y), \tag{20a}$$
$$W(x, y) = W_x(x) W_y(y), \tag{20b}$$
$$\sigma(x, y, \eta) = \sigma_x(x) \sigma_y(y) \sigma_\eta(\eta), \tag{20c}$$
$$\theta(x, y) = \theta_x(x) \theta_y(y), \tag{20d}$$

where

$$\lambda_y(i/n) = \lambda_i, \tag{21a}$$
$$W_y(i/n) = w_i, \tag{21b}$$
$$\theta_y(i/n) = \vartheta_i, \tag{21c}$$
$$\sigma_y(i/n) = s_{1,i}, \tag{21d}$$
$$\sigma_\eta(i/n) = s_{2,i}, \tag{21e}$$

for all $i = 1, \dots, n$. Now, we can identify additional conditions that have to be satisfied by the parameters appearing in (20), in order to guarantee the existence of a separable, closed-form solution to the continuum kernel equations (4), (6). This is formulated in the following proposition. The motivation for the proposition is that we can impose corresponding conditions on the parameters of the large-scale kernel equations (1), (3), so that the closed-form continuum solution to (4), (6) provides an approximation for the solution to (1), (3). This is discussed in detail after Proposition 2 in Remark 2.

Proposition 2. *Let Assumption 1 hold so that* μ *is constant and the continuum parameters are separable as in* (20)*, satisfying* (21)*. Additionally assume that there exists a constant* c_x *satisfying*

$$c_x = \mu \sigma_x(0) \frac{\sigma_\eta(y)}{\theta_y(y)} \int_0^1 \frac{\sigma_y(\eta) \theta_y(\eta)}{\lambda(\eta) + \mu} d\eta + \frac{\mu \lambda(y)}{\lambda(y) + \mu} \frac{\theta_x'(0)}{\theta_x(0)} + \theta_x(0) \int_0^1 \lambda(\eta) q(\eta) \frac{\theta_y(\eta)}{\lambda(\eta) + \mu} d\eta, \tag{22}$$

independently of y*, and a related function* f *satisfying*

$$f(\xi) = \sigma_x(\xi) \frac{\sigma_\eta(y)}{\theta_y(y)} \int_0^1 \frac{\sigma_y(\eta) \theta_y(\eta)}{\lambda(\eta) + \mu} d\eta - \frac{c_x}{\mu} + \frac{\lambda(y)}{\lambda(y) + \mu} \frac{\theta_x'(\xi)}{\theta_x(\xi)}, \tag{23}$$

[7] Note that there is quite a bit of freedom in constructing the continuum approximation, stemming from the fact that the only restriction in obtaining the continuum parameters from the parameters of the $n + 1$ system is (7) (see also [17, 18] for details).

[8] We note that these are merely sufficient conditions, so that it may be possible to derive closed-form solutions under different assumptions as well. In particular, as opposed to Assumption 1, one may consider spatially-varying $\lambda_i, i = 1, \dots, n$ and μ. However, deriving a closed-form solution under such conditions likely requires imposing additional assumptions on the parameters, as opposed to the three conditions (22)–(24) we impose in Proposition 2. Regardless, the case of constant transport speeds λ_i, μ appears at least in the examples considered in, e.g., [2, Sect. VII] and [3, Sect. VI], while such an assumption holds for linearized hyperbolic systems, describing, e.g., traffic or shallow water flows, around a constant equilibrium point; see, e.g., [4, 24].

for all $\xi \in [0,1]$, independently of y, and

$$f'(\xi) = W_x(\xi)\theta_x(\xi) \int_0^1 \frac{W_y(y)\theta_y(y)}{\lambda(y)+\mu} dy, \tag{24}$$

for all $\xi \in [0,1]$. Then, the solution to the continuum kernel equations (4), (6) is given by

$$k(x,\xi,y) = -\exp\left(\frac{c_x}{\mu}x\right)\exp\left(-\frac{c_x}{\mu}\xi\right)\theta_x(\xi)\frac{\theta_y(y)}{\lambda(y)+\mu}, \tag{25a}$$

$$\bar{k}(x,\xi) = \exp\left(\frac{c_x}{\mu}x\right) f(\xi) \exp\left(-\frac{c_x}{\mu}\xi\right). \tag{25b}$$

Proof. The proof can be found in Appendix A. □

Remark 1. If λ is assumed to be constant, it is possible to make the somewhat implicit conditions of Proposition 2 explicit. In this case, the existence of the constant c_x and the function f reduces to the existence of a constant c_y satisfying

$$c_y = \frac{\sigma_\eta(y)}{\theta_y(y)} \int_0^1 \sigma_y(\eta)\theta_y(\eta) d\eta, \tag{26}$$

independently of y, i.e., either there exists a constant c such that $\sigma_\eta = c\theta_y$ or the integral appearing in (26) is zero. If such c_y exists, the constant c_x is then given by

$$c_x = \frac{\mu}{\lambda+\mu}\left(c_y\sigma_x(0) + \lambda\frac{\theta_x'(0)}{\theta_x(0)} + \frac{\lambda}{\mu}\theta_x(0)\int_0^1 q(y)\theta_y(y)dy\right), \tag{27}$$

and the function f is given by

$$f(\xi) = \frac{c_y}{\lambda+\mu}\sigma_x(\xi) - \frac{c_x}{\mu} + \frac{\lambda}{\lambda+\mu}\frac{\theta_x'(\xi)}{\theta_x(\xi)}. \tag{28}$$

Thus, the condition (24) reduces to

$$c_y\sigma_x'(\xi) + \lambda\frac{\theta_x''(\xi)\theta_x(\xi) - \theta_x'(\xi)^2}{\theta_x(\xi)^2} = W_x(\xi)\theta_x(\xi)\int_0^1 W_y(y)\theta_y(y)dy, \tag{29}$$

for all $\xi \in [0,1]$.

Remark 2. The special case of constant λ in the continuum translates to every λ_i being the same for all $i = 1,\dots,n$. Similarly, the other assumptions considered in Remark 1 can be translated to conditions on the parameters of the large-scale equations shown in (19). Regarding (26), we either need that there exists a constant c such that $\vartheta_i = cs_{2,i}$ for all $i = 1,2,\dots,n$, or

$$\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^n s_{1,i}\vartheta_i = 0, \tag{30}$$

where the limit converges to the integral[9] in (26), by construction of the continuum parameters. Similarly, the condition (29) can be written in terms of (19) by replacing the integral with the corresponding infinite sum, i.e.,

$$c_y\sigma_x'(\xi) + \lambda\frac{\theta_x''(\xi)\theta_x(\xi) - \theta_x'(\xi)^2}{\theta_x(\xi)^2} = W_x(\xi)\theta_x(\xi)\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^n w_i\vartheta_i, \tag{31}$$

for all $\zeta \in [0,1]$.

Remark 3. Even though it is reasonable for one to think that Assumption 1 together with the assumptions in Remark 2 lead to closed-form solutions for the exact kernels PDEs (1), (3), corresponding to the $n+1$ system, as well, this is not the case. For example, in more technical terms, even when $\int_0^1 \sigma_y(y)\theta_y(y)dy = 0$, which corresponds to $c_y = 0$ in (26), this does not imply that the respective sum in (30) would necessarily be zero for any finite n (it is zero, but only in the limit $n \to \infty$). In fact, such types of continuum approximations (e.g., of sums by integrals) is the key to obtain a closed-form solution to the continuum kernel PDE (in contrast to the exact kernels PDEs), which even though results in applying to the $n+1$ system approximate control kernels, it remains stabilizing in closed loop. In particular, as we have shown in [17, 18], an approximate solution $(k_{\mathrm{a}}^i)_{i=1}^{n+1}$ to (1), (3) based on the solution to (4), (6) is given by

$$k_{\mathrm{a}}^i(x,\xi) = k(x,\xi,i/n), \qquad i = 1,2,\dots,n, \tag{32a}$$

$$k_{\mathrm{a}}^{n+1}(x,\xi) = \bar{k}(x,\xi). \tag{32b}$$

Remark 4. The conditions in Proposition 2 or Remark 1 may be restrictive (although it is straightforward for one to verify them, so to obtain a closed-form solution without having to undergo any power series-based or other approximation for computing the solution to (4), (6)). However, whenever such an explicit solution is possible to find and, in fact, we identify a class of systems when this is possible via Proposition 2 and Remark 1, it significantly reduces computational complexity. This is particularly useful in the case in which the explicit solution to the continuum kernels equations (4), (6) is employed for computation of stabilizing kernels for the large-scale system (B.1), (B.2) corresponding to the $n+1$ kernels equations (1), (3). For example, as also illustrated in [17, Sect. V] (see also [18, Sect. V]) for the numerical example taken from [2], it is possible to stabilize a large-scale $n+1$ system counterpart, for arbitrarily large n, with almost trivial computations, thanks to the availability of a closed-form solution to the continuum kernel equations (but not to the respective $n+1$

[9] Formally, to the Riemann integral, and due to continuity of the parameters, to the standard Lebesgue integral.

kernels equations). In fact, this approach (of computing stabilizing kernels for large-scale systems via continuum approximation) is particularly useful exactly when an explicit solution to the $n+1$ kernels equations (1), (3) is not available, but an explicit solution is available for the respective continuum equations (4), (6) (as is the case with the example from [2]).

Furthermore, because the conditions in Proposition 2 concern the continuum parameters, there is some flexibility degree for their satisfaction in the case in which the continuum parameters involved (corresponding to the continuum system (B.4), (B.5)) are obtained as continuum approximation (which may not be unique) of the respective sequences of parameters of the large-scale system counterpart (corresponding to system (B.1), (B.2)). This is discussed in detail in [17, Sect. V]. In particular, the parameters of (4), (6) have to be tied to the parameters of (1), (3) in some way, which, in the present case, happens through (7), such that the solution to (4), (6) can approximate the solution to (1), (3). One instance where these conditions may be satisfied is illustrated in Example 1. Another usefulness of having identified a class of equations (4), (6) that feature a closed-form solution stems from the fact that such closed-form solutions provide exact solutions that can be used for computation of exact errors of power series-based approximations, as it is done in Section 5.

Example 1. The simplest case in which the conditions of Remark 1 are satisfied is when $c_y = 0$ and both sides of (29) are zero. Moreover, the solution formula becomes simpler in the particular case $c_x = 0$. These particular conditions (and particular value of c_x) are met in the example considered in [2, Sect. VII], where $\lambda = \mu = 1, W_y(y) = \sigma_y(y) = y - \frac{1}{2}, \theta_y(y) = y(y-1), \theta_x(x) = -70\exp\left(\frac{35}{\pi^2}x\right)$, and $q(y) = \cos(2\pi y)$, in which case the solution (25) becomes

$$k(x,\xi,y) = 35y(y-1)\exp\left(\frac{35}{\pi^2}\xi\right), \tag{33a}$$

$$\bar{k}(x,\xi) = \frac{35}{2\pi^2}. \tag{33b}$$

The other parameter functions are $\sigma_\eta = \sigma_y$, $\sigma_x(x) = x^3(x+1)$, and $W_x(x) = x(x+1)e^x$, although they do not appear in the expression of the solution as $c_x = c_y = 0$ and $\int_0^1 W_y(y)\theta_y(y)dy = 0$.

5. Numerical Experiments: Power Series-Based Computation for Example 1

We test the power series algorithm (full- and reduced-order in y) with the parameters of Example 1 to get an idea of the practical computational requirements and the accuracy of the method. The numerical experiments are performed in MATLAB 2023b on a 2023 MacBook Pro with Apple M3 chip and 8 GB of memory. In Table 2, numerical results are presented of computation of the full-order power series (12), where the first column shows the order N of the approximation; the second column shows the number of unknown coefficients (both K^{a} and $\bar{K}^{\mathrm{a}}$ together); the third column shows the number of (linear) equations (that have to be solved for determining the series coefficients); the fourth column shows the computational time (in seconds); the fifth column shows the residual of the least-squares fit when the set of linear equations is solved; and the sixth column shows the maximal absolute error between the computed and the exact, continuum control gains (i.e., the kernels (33) evaluated at $x = 1$). One can see that both the number of unknowns and the number of equations grow rapidly along with N, and hence, so does the computational time. On the flip side, the residual of the least-squares fit and the maximal error of the control kernel approximation become smaller as N increases. However, for larger values of N, (notable) improvements in the approximation accuracy are only obtained with even N, e.g., the max error only reduces by 0.046 when comparing $N = 14$ and $N = 15$, whereas for $N = 15$ and $N = 16$ the max error reduces by 0.506. This may be related to approximating the even function $q(y) = \cos(2\pi y)$, as most of the other parameters (apart from the exponential terms in θ and W) are polynomials, for which the truncated Taylor series are exact.

N	#K	#Eq	C. time	$\|\mathbf{Ax}-\mathbf{b}\|_2$	max error
12	546	1082	11.78 s.	1.90	16.95
13	665	1285	13.71 s.	0.669	8.79
14	800	1510	18.27 s.	0.209	0.668
15	952	1758	25.24 s.	$5.53\cdot10^{-2}$	0.622
16	1122	2030	32.79 s.	$1.13\cdot10^{-2}$	0.116
17	1311	2327	40.90 s.	$3.10\cdot10^{-3}$	$9.14\cdot10^{-2}$
18	1520	2650	50.35 s.	$6.83\cdot10^{-4}$	$7.23\cdot10^{-3}$
19	1750	3000	62.33 s.	$1.42\cdot10^{-4}$	$7.61\cdot10^{-3}$
20	2002	3378	77.70 s.	$2.82\cdot10^{-5}$	$5.68\cdot10^{-4}$

Table 2: Numerical results for full-order (in y) power series (12) corresponding to Example 1.

Table 3 shows the corresponding results as Table 2 but with reduced order $N_y = 2$ for $k(x,\xi,y)$ in (18). No truncation has been applied to the parameter approximations (13), although the parameters other than q are inherently low-order polynomials in y, so that the higher powers of y do not appear in the Taylor series approximations anyway. In fact, from (4) and (6a) if follows that, for the parameters of Example 1, only powers of y of order $N_y = 2$ appear in (18). Thus, $N_y = 2$ in the present case provides an exact approximation.[10] Consequently, one can see that the number of unknowns in Table 3 grows much slower than in Table 2, while the number of equations grows slower as

[10] In this case, we, in fact, know a priori that $N_y = 2$ is exact, because the closed-form solution (33) is a second-order polynomial in y.

well (although not as drastically). These lower numbers of unknowns and equations also reflect the slower increase in computational time as N increases. When it comes to accuracy, both the residual of the least-squares fit and the maximal absolute error to the exact control gain decrease as N increases, and notable improvements in accuracy are obtained for even N, similarly to Table 2. Since the reduced-order power-series (18) is different from the full-order one, there are some discrepancies between the residuals and maximal errors when N is small. Nevertheless, both approximations appear to converge to the exact solution at the same rate as N increases.

N	#K	#Eq	C. time	$\|\mathbf{Ax}-\mathbf{b}\|_2$	max error
12	326	862	7.23 s.	1.93	16.17
13	379	999	8.85 s.	0.673	8.32
14	436	1146	10.47 s.	0.210	0.510
15	497	1303	12.61 s.	$5.55 \cdot 10^{-2}$	0.658
16	562	1470	14.74 s.	$1.34 \cdot 10^{-2}$	0.110
17	631	1647	17.06 s.	$3.10 \cdot 10^{-3}$	$9.08 \cdot 10^{-2}$
18	704	1834	19.66 s.	$6.84 \cdot 10^{-4}$	$7.27 \cdot 10^{-3}$
19	781	2031	22.96 s.	$1.42 \cdot 10^{-4}$	$7.61 \cdot 10^{-3}$
20	862	2238	26.07 s.	$2.82 \cdot 10^{-5}$	$5.68 \cdot 10^{-4}$

Table 3: Numerical results of computation of reduced-order (in y) power series (18) with $N_y = 2$ corresponding to Example 1.

As the parameter q only appears in an integral in the boundary condition (6b), it may not in fact be necessary to approximate it by a power series, as long as the integral can be evaluated in closed form. Considering that k in the integral is approximated by power series (and λ is constant), the integral can generally be evaluated in closed form by using integration by parts, as long as the integral of q over the unit interval can be computed in closed form. This is demonstrated in Table 4, which is analogous to Table 3, except that the exact expression $q(y) = \cos(2\pi y)$ is used in the computations instead of a power series approximation. The data of Table 4 is consistent with the data of Table 3, except that for larger values of N the maximal error to the exact solution is one order of magnitude smaller than with the power series approximation. Considering that this can be achieved in virtually the same computational time, it may be preferable to use the exact expression of q in the computations (when possible) instead of a power series approximation.

N	#K	#Eq	C. time	$\|\mathbf{Ax}-\mathbf{b}\|_2$	max error
12	326	862	5.27 s.	2.04	13.14
13	379	999	6.66 s.	0.712	5.34
14	436	1146	7.80 s.	0.207	1.16
15	497	1303	13.31 s.	$5.44 \cdot 10^{-2}$	0.159
16	562	1470	16.17 s.	$1.34 \cdot 10^{-2}$	$2.44 \cdot 10^{-2}$
17	631	1647	18.94 s.	$3.11 \cdot 10^{-3}$	$4.16 \cdot 10^{-3}$
18	704	1834	19.37 s.	$6.84 \cdot 10^{-4}$	$7.42 \cdot 10^{-4}$
19	781	2031	23.23 s.	$1.42 \cdot 10^{-4}$	$1.32 \cdot 10^{-4}$
20	862	2238	27.80 s.	$2.82 \cdot 10^{-5}$	$2.27 \cdot 10^{-5}$

Table 4: Numerical results using the exact q function, of reduced-order (in y) power series (18) with $N_y = 2$ corresponding to Example 1.

6. Application to Stabilization of a Large-Scale System

6.1. Parameters and Their Continuum Approximation

We consider a large-scale system of $n+1$ hyperbolic PDEs shown in Appendix B (see [3]) with parameters

$$\lambda_i(x) = 1, \tag{34a}$$
$$\mu(x) = 1, \tag{34b}$$
$$\sigma_{i,j}(x) = x^3(x+1)\left(\frac{i}{n}-1\right)\left(\frac{j}{n}-1\right), \tag{34c}$$
$$W_i(x) = 2x(x+1)\frac{i}{n}, \tag{34d}$$
$$\theta_i(x) = -70x\frac{i}{n}\left(\frac{i}{n}-1\right), \tag{34e}$$

for $i, j = 1, 2, \ldots, n$, where we take $n = 10$, and $q = [-0.127, -0.119, -0.197, -0.28, -0.272, -0.235, -0.164, -0.113, -0.124, 0.047]$[11]. As discussed in Section 2, when n is sufficiently large, such $n+1$ system can be approximated by a corresponding ensemble of linear PDEs, shown in Appendix B (see [2, 17]). This also implies that the solution to the large-scale kernel equations (1), (3) can be approximated by solving the corresponding continuum kernel equations (4), (6) (see [17] for details). In order to do this, we need to first construct the continuum parameters such that the relations (7) are satisfied. Due to the structure of the parameters (34), this can be achieved by replacing i/n and j/n by y and η, respectively, which results in continuum parameters as

$$\lambda(x, y) = 1, \tag{35a}$$
$$\mu(x) = 1, \tag{35b}$$
$$\sigma(x, y, \eta) = x^3(x+1)(y-1)(\eta-1), \tag{35c}$$
$$W(x, y) = 2x(x+1)y, \tag{35d}$$
$$\theta(x, y) = -70xy(y-1). \tag{35e}$$

[11] The values of q are generated by sampling the polynomial $y(y-1)$ at $y = 0.1, 0.2, \ldots, 1$ and then perturbing the data with evenly-distributed random numbers from the open interval $(-0.05, 0.05)$.

For the q_i data, we construct a continuum approximation by utilizing the `polyfit` routine in MATLAB. We try polynomials of orders $M = 2, \dots, 6$, which are illustrated in Figure 1. Note that we have chosen to locate the q_i data at points $y = i/n$ for $i = 1, \dots, n$, meaning that the leftmost data point is located at $1/n = 1/10$.[12] It can be seen that the fits of orders $M = 2, 3, 4$ do not notably differ from one another, while the higher-order ones for $M = 5, 6$ provide a different, nevertheless mutually similar fit. We proceed with the lowest order fit $M = 2$ to the power series computations, as that already seems to provide a sufficiently accurate approximation for the q_i data. As we demonstrate at the end of the next section (see Table 7), this choice has only a marginal effect to the numerical results presented in the next section, as all the polynomial fits presented in Figure 1 are close to each other in the L^2 sense.

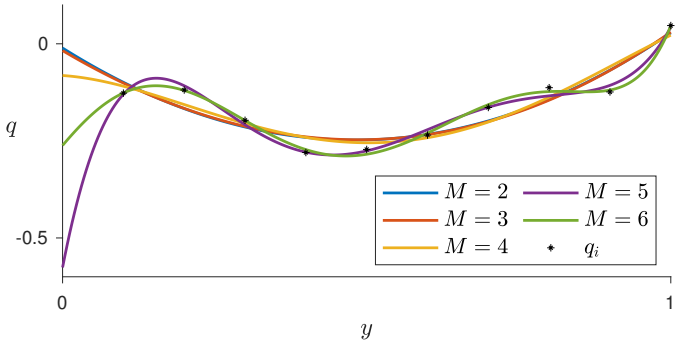

Figure 1: Polynomial fits of order $M = 2, \dots, 6$ to the q_i data.

6.2. Computation of Continuum Kernels with Power Series

We note first that the continuum parameters constructed in Section 6.1 are separable, low-order polynomials in the spatial variables. However, as $\theta_y(y) = y(y-1)$ and $\sigma_y(y) = \sigma_\eta(y) = y - 1$, we have $\dfrac{\sigma_\eta(y)}{\theta_y(y)} = \dfrac{1}{y}$ and $\displaystyle\int_0^1 \sigma_y(\eta)\theta_y(\eta)d\eta = \frac{1}{12}$, so that there is no constant c_y satisfying (26), and hence, Proposition 2 is not applicable for finding the solution to (4), (6) in closed form. Instead, we employ the power series approach to solve the continuum kernel equations. Note that we were not able to find a closed-form solution to (1), (3) for parameters (34) either.

Considering that the parameters are low-order polynomials, one may guess that a power series approximation of the same order is sufficient for solving the equations. Hence, we initialize our computation of the kernel equations, with power series approximation of order $N = 6$, as that is the highest order term appearing in the parameters shown in (35), namely $x^4 y\eta$ in $\sigma(x, y, \eta)$. The results are shown in Table 5, which shows the data corresponding to Table 2 for this example. However, as we do not know the exact solution to the kernel equations (4), (6) for parameters (35), instead of the maximal error to the exact solution, the penultimate column shows the maximal difference d_{n+1} to the solution of the large-scale kernel equations, which we have computed based on a finite-difference approximation of (1), (3). Moreover, the last column shows the maximal difference d_{N-1} to the solution obtained with power series of order $N-1$. The last two columns imply that the power series solutions to (4), (6) converge as N increases. However, as $n = 10$, there persists an approximation error between the solutions to (1), (3) and (4), (6), which would tend to zero as $n \to \infty$ (see [17]). Moreover, the second-order polynomial fit to the q_i data also induces some (additional) approximation errors in this case.

The obtained approximate solutions (12a) evaluated at $x = 1$ are displayed in Figure 2, where visually it is difficult to distinguish the solutions after $N = 20$, further supporting the conclusion of convergent power series approximations. However, it is interesting to note that a much higher order approximation is needed for the solution than the powers of the parameters (35), which, in this case, are represented exactly by a Taylor series of order six. In fact, the convergence rate of the power series approximation appears to be slower than in Example 1 (based on the values of the least squares fit $\|\mathbf{Ax} - \mathbf{b}\|_2$), where the approximation accuracy is good already at the order of around $N = 15$, even though the parameters are not polynomials. Regardless, in Example 1 both the parameters and the solution (33) can be approximated at a satisfactory accuracy with this order of approximation. In general, as we may have no a priori information about the form of the solution, the convergence rate and approximation accuracy need to be assessed based on the error of the least squares fit $\|\mathbf{Ax}-\mathbf{b}\|_2$ as N increases.[13] is, in general, expected that such upper bounds may be conservative for practical computations.

N	#K	#Eq	C. time	$\|\mathbf{Ax} - \mathbf{b}\|_2$	d_{n+1}	d_{N-1}
6	112	221	1.50 s.	4.64	5.90	–
10	352	533	4.74 s.	3.46	4.78	0.161
15	952	1223	15.09 s.	2.07	2.86	0.127
20	2002	2363	36.70 s.	0.414	1.15	0.184
25	3627	4078	89.65 s.	$2.6 \cdot 10^{-2}$	1.09	$2.1 \cdot 10^{-2}$
30	5952	6493	212.29 s.	$9.3 \cdot 10^{-4}$	1.09	$1.6 \cdot 10^{-5}$

Table 5: Numerical results for full-order (in y) power series approximation (12) corresponding to parameters (35).

[12] One may just as well locate the points differently, e.g., at $(i-1)/n$ for $i = 1, \dots, n$ (see [17, Rem. 4.4] for details).

[13] Although here we derive such a convergence rate proxy based on numerical experiments, in principle, an upper bound for the convergence rate of the power series could be derived analytically. However, this would require estimating the derivatives of $(k, \bar{k})$ up to arbitrary order, which, based on the discussion at the end of Section 3.2, may be computationally laborious. Moreover, it

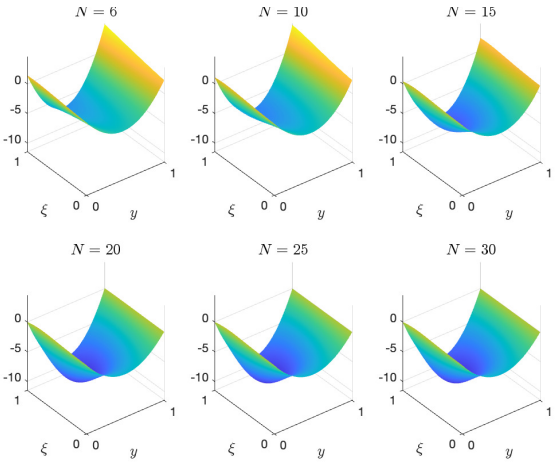

Figure 2: The control gain $k(1, \xi, y)$ approximated by full-order (in y) power series (12a) for $N = 6, 10, 15, 20, 25, 30$.

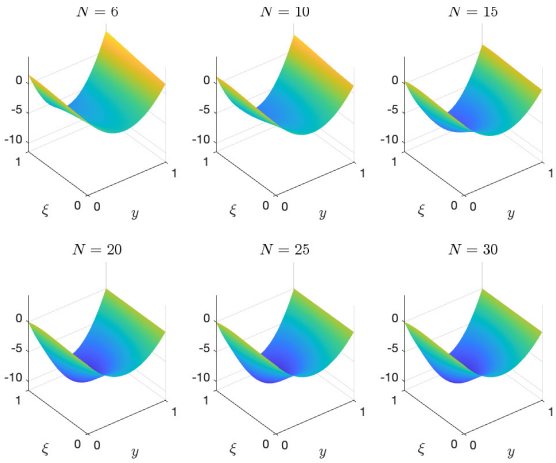

Figure 3: The control gain $k(1, \xi, y)$ approximated by reduced-order (in y) power series (18) for $N = 6, 10, 15, 20, 25, 30$ and $N_y = 2$.

We employ also here the reduced-order approximation in y. Considering that the highest power of y appearing in the parameters is two, we repeat the computations of Table 5 with reduced order $N_y = 2$ in y. Similarly to what we saw in Example 1, the results shown in Table 6 for the reduced-order (in y) power series are analogous to the full-order results in terms of accuracy, apart from some minor discrepancies with smaller values of N. The obtained control gains $k(1, \xi, y)$ for different N are displayed in Figure 3, where one can notice some differences for smaller values of N as compared to Figure 2. For larger values of N (bottom plot of Figure 2), the two figures are virtually identical. Thus, the reduced-order power series provides as good of an approximation (when N is sufficiently large) as the full-order power series, with the notable benefit that the computational complexity (number of unknowns/equations and computational time) increases at a slower rate with respect to N than in the full-order power series approximation, as already reported in Table 1.

N	#K	#Eq	C. time	$\|\mathbf{Ax} - \mathbf{b}\|_2$	d_{n+1}	d_{N-1}
6	92	201	1.37 s.	4.78	5.95	–
10	232	413	3.21 s.	3.62	4.83	0.173
15	497	768	7.37 s.	2.14	2.82	0.138
20	862	1223	13.03 s.	0.417	1.15	0.180
25	1327	1778	21.52 s.	$2.7 \cdot 10^{-2}$	1.09	$2.0 \cdot 10^{-2}$
30	1892	2433	33.98 s.	$9.3 \cdot 10^{-4}$	1.09	$1.9 \cdot 10^{-5}$

Table 6: Numerical results for reduced-order (in y) power series approximation (18) with $N_y = 2$ corresponding to the parameters (35).

To conclude this section, we justify our choice of $M = 2$ for the order of the polynomial fit to the q_i data. The results are displayed in Table 7 for full-order power series (in y) of order $N = 20$, which show that the different orders of polynomial fit to the q_i data have a marginal effect on the results. Thus, choosing the lowest order fit of $M = 2$ is reasonable, as that results in the simplest continuum approximation.

M	#K	#Eq	C. time	$\|\mathbf{Ax} - \mathbf{b}\|_2$	d_{n+1}
2	2002	2363	34.96 s.	0.4136	1.1484
3	2002	2363	35.49 s.	0.4136	1.1484
4	2002	2363	35.78 s.	0.4136	1.1486
5	2002	2363	36.13 s.	0.4140	1.1497
6	2002	2363	36.44 s.	0.4138	1.1490

Table 7: Numerical results for full-order (in y) power series approximation (12) of order $N = 20$ corresponding to parameters (35) with polynomials of order $M = 2, \ldots, 6$ fitted to the q_i as displayed in Figure 1.

6.3. Stabilization Using Power Series-Based Approximate Continuum Kernels

We simulate the $n + 1$ system with parameters (34). The $n + 1$ hyperbolic PDE system, is approximated using finite differences with 256 grid points in x. For the backstepping control law, we employ the continuum kernels computed in Section 6.2 for different orders of approximation N. Note that with parameters (35), it is verified in simulation that the open-loop system is unstable. The employed hardware and software are the same as in Section 5, and the data displayed in Figures 4 and 5 takes about 14 seconds to compute by solving the finite-difference approximation of the closed-loop system with the `ode45` solver in MATLAB.

Figure 4 shows the control inputs corresponding to the continuum kernels computed using the full-order (in y) power series of order $N = 6, 10, 15, 20, 25$ and the solution[14] to the $n + 1$ kernel equations (1), (3). The controls

[14]The solution is computed based on a finite-difference approximation of (1), (3).

start to diverge for $N = 6$, implying that the controls fail to stabilize the closed-loop system. This is expected based on the numerical experiments of Section 6.2, as the accuracy obtained with such a low-order approximation is not adequate. As the approximation order N increases, one can see that the controls tend to zero, implying that, for larger values of N, the controls are stabilizing. For the lower orders $N = 10$ and $N = 15$ the controls are distinguishable, and the decay rate is relatively slow, whereas for $N = 20$ and $N = 25$ the controls are virtually identical and converge to zero after two seconds, implying that all the state components have converged to zero by that time as well. This is consistent with the control gains displayed in Figure 2, which are distinctly different for $N = 5, 10, 15$ and virtually identical for $N = 20, 25, 30$. Finally, we see that the approximate controls for $N = 20$ and $N = 25$ are also very close to the controls computed based on the solution to the $n+1$ kernel equations (1), (3).

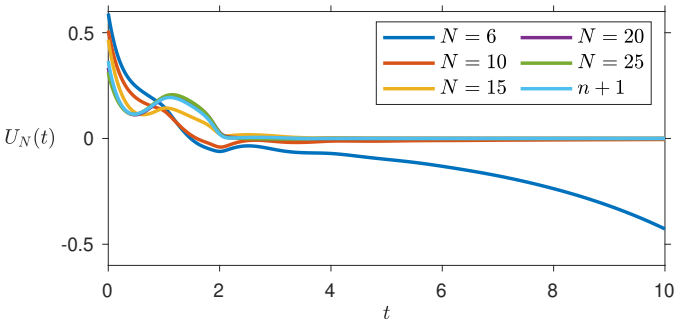

Figure 4: The controls obtained with approximate continuum kernels computed with full-order (in y) power series (12) of order $N = 6, 10, 15, 20, 25$ and by solving (1), (3) for the $n + 1$ kernels.

Figure 5 shows the corresponding results as Figure 4 but for the reduced-order (in y) power series approximation (18) for the continuum kernel. The same comments apply to Figure 5 as to Figure 4. A slight difference can be observed for the low-order approximations, where the case $N = 6$ diverges at a slower rate than in Figure 4 (the closed-loop system is still unstable), while the case $N = 10$ appears to converge to zero at a faster rate. Regardless, for larger values of N, the controls are virtually indistinguishable from one another, as well as from the ones displayed in Figure 4. This is consistent with the control gains in Figure 3. We note here that as [17, Thm 4.1] predicts, for sufficiently large n ($n = 10$ in the present case), the corresponding feedback law obtained employing the continuum control kernel $k(1, \xi, y)$ is stabilizing, provided that $k(1, \xi, y)$ is approximated at a sufficient accuracy via the power series. In turn, the latter is established in Theorem 1. In conclusion, stabilization with performance close to the one corresponding to exact control kernels is achieved, with computational complexity of order $\mathcal{O}(N^2)$ that does not grow with n, in contrast to the computation of the exact control kernels (see Table 1). We note that, the computational complexity of solving (1), (3) using a finite-difference approximation grows (roughly linearly) with n, as shown in [17, Figure 6].

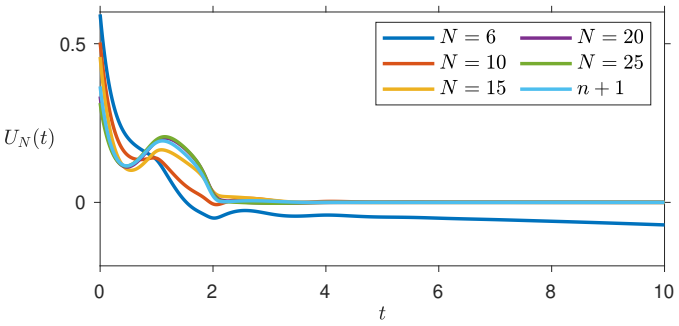

Figure 5: The controls obtained with approximate continuum kernels computed with reduced-order (in y) power series (18) of order $N = 6, 10, 15, 20, 25$ and $N_y = 2$ and by solving (1), (3) for the $n + 1$ kernels.

7. Conclusions and Future Work

In this paper, we provided computational tools for construction of backstepping-based stabilizing kernels for continua of hyperbolic PDEs, and thus, also for large-scale PDE systems. These tools include both explicit kernels construction (even though the class of systems for which this is possible may be restrictive) and approximation via power series. We demonstrated the accuracy and efficiency of these tools on two numerical examples. In the first, we derived the closed-form solution to continuum kernel equations and demonstrated the convergence of the power series approximation to the closed-form solution as N increases. In the second numerical example, we could not derive the solution in closed form, and thus, we employed only the power series approach (and reduced-order power series approach) to solve the continuum kernel equations. We further demonstrated that the computed continuum kernels provided a stabilizing feedback law for the corresponding large-scale system of linear PDEs, with performance comparable to the case of employing the exact, large-scale control kernels.

For future work, the efficacy of Algorithm 1 could, potentially, be substantially improved by adopting the double vector-matrix framework from [21, Sect. II.B]. However, this has to be modified to fit the triple power series employed here, potentially requiring a triple vector-matrix framework, which may not be straightforward to implement. Another topic for future research is to derive analytically the convergence rate of the power series representation for the solution to the continuum kernel equations, thus providing explicit (although potentially conservative) estimates of the lowest power required for an accurate, power series-based approximation (given a desired approximation error). However, as discussed at the end of Section 3.2, derivation of such bounds is of the same computational complexity $\mathcal{O}(N^3)$ as the proposed power series procedure for actually computing the kernels. Hence, the

least-squares error of Algorithm 1 may be the most practical option for assessing the accuracy of the computed solution.

Appendix

Appendix A. Proof of Proposition 2

The solution stated in Proposition 2 can be constructed by looking for a separable solution to (4), (6), i.e., $k(x,\xi,y) = k_x(x)k_\xi(\xi)k_y(y)$ and $\bar{k}(x,\xi) = \bar{k}_x(x)\bar{k}_\xi(\xi)$. After dividing the equations (4) by the solution, we get

$$\mu\frac{k_x'(x)}{k_x(x)} - \lambda(y)\frac{k_\xi'(\xi)}{k_\xi(\xi)} - \frac{\theta_x(\xi)\theta_y(y)\bar{k}_x(x)\bar{k}_\xi(\xi)}{k_x(x)k_y(y)k_\xi(\xi)} = \frac{\sigma_x(\xi)\sigma_\eta(y)}{k_y(y)}\int_0^1 \sigma_y(\eta)k_y(\eta)d\eta, \tag{A.1a}$$

$$\mu\frac{\bar{k}_x'(x)}{\bar{k}(x)} + \mu\frac{\bar{k}_\xi'(\xi)}{\bar{k}_\xi(\xi)} = \frac{W_x(\xi)k_x(x)k_\xi(\xi)}{\bar{k}_x(x)\bar{k}_\xi(\xi)}\int_0^1 W_y(y)k_y(y)dy, \tag{A.1b}$$

with boundary conditions

$$k_x(x)k_\xi(x)k_y(y) = -\frac{\theta_x(x)\theta_y(y)}{\lambda(y)+\mu}, \tag{A.2a}$$

$$\mu\bar{k}_x(x)\bar{k}_\xi(0) = k_x(x)k_\xi(0)\int_0^1 \lambda(y)q(y)k_y(y)dy. \tag{A.2b}$$

As the boundary conditions have to hold for all $x, y \in [0,1]$, the second boundary condition implies that $\bar{k}_x(x)/k_x(x)$ has to be constant. Inserting this into (A.1a), we also get that $\mu k_x'(x)/k_x(x)$ has to be constant, meaning that we can set $k_x(x) = \exp\left(\frac{c_x}{\mu}x\right) = \bar{k}_x(x)$ for some c_x, which eventually turns out to be given by (22).

The first boundary condition (A.2a) gives

$$(\lambda(y)+\mu)\frac{k_y(y)}{\theta_y(y)} = -\frac{\theta_x(x)}{k_x(x)k_\xi(x)}, \tag{A.3}$$

meaning that we can set

$$k_y(y) = c_1\frac{\theta_y(y)}{\lambda(y)+\mu}, \tag{A.4}$$

for some $c_1 \neq 0$, and further get

$$k_\xi(x) = -\frac{1}{c_1}\frac{\theta_x(x)}{k_x(x)} = -\frac{1}{c_1}\theta_x(x)\exp\left(-\frac{c_x}{\mu}x\right), \tag{A.5}$$

which, in combination with (A.3), gives (25a). Now, substituting (25a) and $\bar{k}_x(x) = \exp\left(\frac{c_x}{\mu}x\right)$ to (A.1a), we obtain

$$c_x - \lambda(y)\left(\frac{\theta_x'(\xi)}{\theta_x(\xi)} - \frac{c_x}{\mu}\right) + (\lambda(y)+\mu)\exp\left(\frac{c_x}{\mu}\xi\right)\bar{k}_\xi(\xi) = \sigma_x(\xi)(\lambda(y)+\mu)\frac{\sigma_\eta(y)}{\theta_y(y)}\int_0^1 \frac{\sigma_y(\eta)\theta_y(\eta)}{\lambda(\eta)+\mu}d\eta, \tag{A.6}$$

which gives $\bar{k}_\xi(\xi) = f(\xi)\exp\left(-\frac{c_x}{\mu}\xi\right)$ with f given in (23). Now, multiplying (A.1b) by $\bar{k}_\xi(\xi)$ and using

$$\bar{k}_\xi'(\xi) = -\frac{c_x}{\mu}\bar{k}_\xi(\xi) + f'(\xi)\exp\left(-\frac{c_x}{\mu}\xi\right), \tag{A.7}$$

we get

$$f'(\xi)\exp\left(-\frac{c_x}{\mu}\xi\right) = W_x(\xi)\theta_x(\xi)\exp\left(-\frac{c_x}{\mu}\xi\right)\int_0^1 \frac{W_y(y)\theta_y(y)}{\lambda(y)+\mu}dy, \tag{A.8}$$

which is satisfied as (24) holds by assumption. Finally, from the second boundary condition (A.2b) we get

$$f(0) = -\frac{\theta_x(0)}{\mu}\int_0^1 \lambda(y)q(y)\frac{\theta_y(y)}{\lambda(y)+\mu}dy, \tag{A.9}$$

which, when combined with (23), gives the stated expression (22) for c_x. This concludes the proof.

Appendix B. Large-Scale System of $n+1$ Hyperbolic PDEs and Its Continuum Approximation

By an $n+1$ system, or a large-scale system, we refer to the following system of $n+1$ linear hyperbolic PDEs

$$u_t^i(t,x) + \lambda_i(x)u_x^i(t,x) = \frac{1}{n}\sum_{j=1}^n \sigma_{i,j}(x)u^j(t,x) + W_i(x)v(t,x), \tag{B.1a}$$

$$v_t(t,x) - \mu(x)v_x(t,x) = \frac{1}{n}\sum_{j=1}^n \theta_j(x)u^j(t,x), \tag{B.1b}$$

with boundary conditions

$$u^i(t,0) = q_i v(t,0), \qquad v(t,1) = U(t), \tag{B.2}$$

for $i = 1,2,\ldots,n$. It follows from [3, Thm 3.2] (see also [17, Sect. II]) that the system (B.1), (B.2) is exponentially stabilizable by a state-feedback law of the form

$$U(t) = \int_0^1 \left[\frac{1}{n}\sum_{i=1}^n k^i(1,\xi)u^i(t,\xi) + k^{n+1}(1,\xi)v(t,\xi)\right]d\xi, \tag{B.3}$$

where $\left(k^i\right)_{i=1}^{n+1}$ is the solution to (1), (3). As already discussed in Section 2, the $n+1$ kernel equations (1), (3) can be approximated by the corresponding continuum kernel equations (4), (6), provided that the continuum parameters are constructed appropriately (see [17, Thm 4.1]). For example, they can be constructed such that the relations in (7) are satisfied. Under an appropriate continuum approximation of the parameters of the $n+1$ system, provided that n is sufficiently large, the control gains in (B.3) can be replaced by the corresponding continuum gains given in (32), with preservation of exponential stability [17, Thm 4.1].

The continuum kernel equations (4), (6) first appeared within the framework of backstepping control design for an ensemble (or continuum) of hyperbolic PDEs of the form

$$u_t(t,x,y)+\lambda(x,y)u_x(t,x,y)=\int_0^1\sigma(x,y,\eta)u(t,x,\eta)d\eta + W(x,y)v(t,x), \quad \text{(B.4a)}$$

$$v_t(t,x)-\mu(x)v_x(t,x)=\int_0^1\theta(x,y)u(t,x,y)dy, \quad \text{(B.4b)}$$

with boundary conditions

$$u(t,0,y)=q(y)v(t,0), \qquad v(t,1)=U(t), \qquad \text{(B.5)}$$

for almost every $y\in[0,1]$, where y is the ensemble variable. The stabilizing backstepping control law for the continuum system (B.4), (B.5) is given by [2, Thm 1]

$$U(t)=\int_0^1\left[\int_0^1 k(1,\xi,y)u(t,\xi,y)dy+\bar{k}(1,\xi)v(t,\xi)\right]d\xi, \quad \text{(B.6)}$$

where $(k,\bar{k})$ is the solution to (4), (6). In fact, for large n, the solutions to (B.1), (B.2) converge to the solutions to (B.4), (B.5) as well, under appropriate conditions [17, Thm 6.1].

References

[1] M. Krstic, A. Smyshlyaev, Boundary Control of PDEs: A Course on Backstepping Designs, SIAM, 2008.

[2] V. Alleaume, M. Krstic, Ensembles of hyperbolic PDEs: Stabilization by backstepping, IEEE Trans. Automat. Control (2024) 1–16.

[3] F. Di Meglio, R. Vazquez, M. Krstic, Stabilization of a system of $n+1$ coupled first-order hyperbolic linear PDEs with a single boundary input, IEEE Trans. Automat. Control 58 (12) (2013) 3097–3111.

[4] H. Yu, M. Krstic, Output feedback control of two-lane traffic congestion, Automatica 125 (2021) Paper No. 109379.

[5] J. Auriol, D. Bresch-Pietri, Robust state-feedback stabilization of an underactuated network of interconnected $n+m$ hyperbolic PDE systems, Automatica 136 (2022) Paper No. 110040.

[6] J. Redaud, J. Auriol, S.-I. Niculescu, Output-feedback control of an underactuated network of interconnected hyperbolic PDE-ODE systems, Syst. Control Lett. 154 (2021) Paper No. 104984.

[7] J.-M. Coron, R. Vazquez, M. Krstic, G. Bastin, Local exponential H^2 stabilization of a 2×2 quasilinear hyperbolic system using backstepping, SIAM J. Control Optim. 51 (3) (2013) 2005–2035.

[8] L. Hu, F. Di Meglio, R. Vazquez, M. Krstic, Control of homodirectional and general heterodirectional linear coupled hyperbolic PDEs, IEEE Trans. Automat. Control 61 (11) (2016) 3301–3314.

[9] J. Wang, M. Krstic, Event-triggered output-feedback backstepping control of sandwich hyperbolic PDE systems, IEEE Trans. Automat. Control 67 (1) (2022) 220–235.

[10] A. Diagne, M. Diagne, S. Tang, M. Krstic, Backstepping stabilization of the linearized *Saint-Venant-Exner* model, Automatica 76 (2017) 345–354.

[11] R. Vazquez, M. Krstic, Marcum Q-functions and explicit kernels for stabilization of 2×2 linear hyperbolic systems with constant coefficients, Syst. Control Lett. 68 (2014) 33–42.

[12] H. Anfinsen, O. M. Aamo, Adaptive Control of Hyperbolic PDEs, Springer, 2019.

[13] R. Vazquez, G. Chen, J. Qiao, M. Krstic, The power series method to compute backstepping kernel gains: theory and practice, in: IEEE Conference on Decision and Control, 2023, pp. 8162–8169.

[14] J. Auriol, K. A. Morris, F. Di Meglio, Late-lumping backstepping control of partial differential equations, Automatica 100 (2019) 247–259.

[15] J. Qi, J. Zhang, M. Krstic, Neural operators for PDE backstepping control of first-order hyperbolic PIDE with recycle and delay, Syst. Control Lett. 185 (2024) Paper No. 105714.

[16] L. Bhan, Y. Shi, M. Krstic, Neural operators for bypassing gain and control computations in PDE backstepping, IEEE Trans. Automat. Control (2024).

[17] J.-P. Humaloja, N. Bekiaris-Liberis, Stabilization of a class of large-scale systems of linear hyperbolic PDEs via continuum approximation of exact backstepping kernels, arXiv, 2403.19455 (v2), 2024.

[18] J.-P. Humaloja, N. Bekiaris-Liberis, On stabilization of large-scale systems of linear hyperbolic PDEs via continuum approximation of exact backstepping kernels, in: IEEE Conference on Decision and Control, 2024.

[19] P. Ascencio, A. Astolfi, T. Parisini, Backstepping PDE design: a convex optimization approach, IEEE Trans. Automat. Control 63 (7) (2018) 1943–1958.

[20] R. Vazquez, J. Zhang, J. Qi, M. Krstic, Kernel well-posedness and computation by power series in backstepping output feedback for radially-dependent reaction-diffusion PDEs on multidimensional balls, Syst. Control Lett. 177 (2023) Paper No. 105538.

[21] X. Lin, R. Vazquez, M. Krstic, Towards a MATLAB Toolbox to compute backstepping kernels using the power series method, arXiv, 2403.16070, 2024.

[22] W. Rudin, Principles of Mathematical Analysis, 3rd Edition, International Series in Pure and Applied Mathematics, McGraw-Hill Book Co., New York-Auckland-Düsseldorf, 1976.

[23] J. Lebl, Tasty Bits of Several Complex Variables: A whirlwind tour of the subject, 2023.
URL `https://www.jirka.org/scv/scv.pdf`

[24] G. Bastin, J.-M. Coron, Stability and Boundary Sabilization of 1-D Hyperbolic Systems, Birkhäuser/Springer, [Cham], 2016.