mathematics

MDPI

Article

Boundary Control for a Certain Class of Reaction-Advection-Diffusion System

Eduardo Cruz-Quintero and Francisco Jurado *

División de Estudios de Posgrado e Investigación, Tecnológico Nacional de México/I.T. La Laguna, Revolución Blvd. and Instituto Tecnológico de La Laguna Av., Torreón 27000, Mexico; d.ecruzq@correo.itlalaguna.edu.mx

* Correspondence: fjurado@itlalaguna.edu.mx

Received: 7 September 2020; Accepted: 19 October 2020; Published: 22 October 2020

Abstract: There are physical phenomena, involving diffusion and structural vibrations, modeled by partial differential equations (PDEs) whose solution reflects their spatial distribution. Systems whose dynamics evolve on an infinite-dimensional Hilbert space, i.e., infinite-dimensional systems, are modeled by PDEs. The aim when designing a controller for infinite-dimensional systems is similar to that for finite-dimensional systems, i.e., the control system must be stable. Another common goal is to design the controller in such a way that the response of the system does not be affected by external disturbances. The controller design for finite-dimensional systems is not an easy task, so, the controller design for infinite-dimensional systems is even more challenging. The backstepping control approach is a dominant methodology for boundary feedback design. In this work, we try with the backstepping design for the boundary control of a reaction-advection-diffusion (R-A-D) equation, namely, a type parabolic PDE, but with constant coefficients and Neumann boundary conditions, with actuation in one of these latter. The heat equation with Neumann boundary conditions is considered as the target system. Dynamics of the open- and closed-loop solution of the PDE system are validated via numerical simulation. The MATLAB-based numerical algorithm related with the implementation of the control scheme is here included.

Keywords: applied mathematics; backstepping; boundary control; computational methods; distributed parameters system; fluid mechanics; partial differential equations; reaction-advection-diffusion equation

1. Introduction

There are a large number of dynamic systems that can be modeled by partial differential equations (PDEs). Many of these systems can be found in the study of mechanical, electrical, or civil engineering as well as in the study of physics and chemistry sciences since all these disciplines deal with the study of fluid flow, heat transfer, electromagnetic field, beam deflection, acoustics, and many other physical phenomena. In the study of health and social sciences, PDEs can also be used to model spread of diseases and viruses as well as population growth. One of the main features of the phenomena mentioned above is that these are listed as distributed parameters systems (DPSs), which are characterized by having two or more independent variables where the combination of one spatial variable with one temporal variable is the most common. Another feature from the DPSs is that these are infinite-dimensional systems. PDEs can be categorized as of parabolic type, elliptic type and hyperbolic type. Each one of these categories describes the behavior of different phenomena, e.g., parabolic PDEs are mainly used to describe heat transfer systems and diffusion systems, elliptic PDEs are used to describe steady state behavior for systems such as electrostatic fields, and hyperbolic PDEs are associated with vibration or wave phenomena. To get the general solution (dynamics) from a PDE

system, it is necessary to set not only initial conditions but boundary conditions which can be from three types, namely, the so-called Dirichlet, Neumann, and Robin conditions [1].

From control theory, the backstepping methodology can be used to force a finite-dimensional nonlinear dynamical system, mainly modeled by ordinary differential equations (ODEs), to behave like a linear system in a new set of coordinates [2,3]. The backstepping methodology has been extended from control of finite-dimensional ODEs systems to control of infinite-dimensional PDEs systems. Control of PDEs systems can be performed in two ways, depending on the location of sensors and actuators, either in the domain dimension or at the boundary.

One of the first attempts about backstepping control design for linear parabolic PDEs was reported in [4]. The backstepping methodology applied to the boundary control of PDEs, from considering Dirichlet and Neumann boundary value problems on the control of an unstable heat equation, was introduced in [5]. In [6], from the structure for a class of one-dimensional (1-D) linear parabolic partial integro-differential equations (P(I)DEs), with one Neumann boundary condition and control signal applied through either Dirichlet and Neumann boundary actuation, the backstepping methodology was used to establish an explicit expression for the gain kernel, namely the gain kernel P(I)DE system, of the boundary control law. Furthermore, the backstepping approach was extended to the design of inverse optimal controllers. From the gain kernel, explicit solutions for the boundary state feedback laws to the control of an unstable heat equation, a heat equation with destabilizing boundary condition, a reaction-diffusion (R-D) equation with spatially varying coefficient for the reaction term, and for a solid propellant rocket model, were obtained. In [7], the backstepping methodology was translated to the design of state observers with boundary sensing. Adaptive boundary controllers for parabolic PDEs were derived in [8–11]. The adaptive approaches were extended to reaction-advection-diffusion (R-A-D) systems in higher dimensions (2-D and 3-D) and to systems with spatially varying unknown parameters [12,13]. Most of the work cited above was compiled in [12,14]. Adaptive control of linear hyperbolic PDEs following the backstepping (Volterra integral transformation) strategy can be found in [15]. Furthermore, through the use of Volterra and Fredholm integral transformations, the input-to-state stability (ISS) approach has been extended to the control of 1-D PDEs in [16].

Boundary control design for a R-A-D equation via backstepping approach seems like a challenging task. Because of the inclusion of partial derivatives in its structure, the searching for the solution of the gain kernel PDE represents a tedious work when trying with the solution for an integral equation equivalent to the kernel PDE. In our work, we focus our attention in the design of a Neumann stabilizing controller for a R-A-D equation, namely, a linear parabolic PDE, but with constant coefficients and Neumann boundary conditions, with actuation in one of these latter. A heat equation with Neumann boundary conditions is considered as target system. This work is motivated from [14], where the Neumann stabilizing controller design to this class of PDE system is left as an exercise to the reader.

The manuscript is summarized as follows. The description of functions spaces is given in Section 2. In Section 3, the R-A-D PDE with constant parameters and boundary conditions of the Neumann type is shown. The R-A-D equation is reduced through a proper change of variables to a R-D system in order to implement the backstepping methodology for boundary control. In Section 4, once that the original system has been reduced to the form of a R-D equation, a target system is proposed in order to, invoking the Lyapunov approach, guarantee exponential stability. In Section 5, the main idea of the backstepping approach, a Volterra integral transformation in order to map the original PDE system into a reduced one, namely, the target system, is applied. As a result, a new PDE system is obtained in function of the so-called kernel gain. Then, by a change of variables, the solution for the integral equation is obtained. By using the successive approximations method for integral equations, as a result, the value of the kernel gain is obtained at Section 6. Once the kernel gain value is obtained, in Section 7 we back to the original variables getting so the stabilizing controller to be implemented on the boundary actuated from the original system. In Section 8, the numerical algorithm used here, in

order to implement and replicate our work, is given. Simulation results are discussed at Section 9. Conclusions are drawn at the end of the paper.

2. Function Spaces

Let us denote by Ω an open set of $\mathbb{R}^n$ with boundary Γ. Let us assume that either Ω is Lipschitz or Ω is of class C^r for $r \geq 1$. The generic point of $\mathbb{R}^n$ is denoted by $x = \{x_1, \ldots, x_n\}$. The Lebesgue measure on $\mathbb{R}^n$ is denoted by $dx = dx_1 \ldots dx_n$.

Let us denote by $C(\Omega)$ (resp. $C^k(\Omega), k \in \mathbb{N}$ or $k = \infty$) the space of real continuous functions on Ω (resp. the space of k times continuously differentiable functions on Ω), $C(\overline{\Omega})$ (resp. $C^k(\overline{\Omega})$) represents the space of real continuous functions on $\overline{\Omega}$ (resp. the space of k times continuously differentiable functions on Ω). The spaces of real C^∞ functions on Ω with a compact support in Ω are denoted by C_0^∞.

For $p \in [1, \infty)$, $L^p(\Omega)$ is the space of real functions on Ω which are L^p for the Lebesgue measure. It is a Banach space for the norm

$$\|u\|_{L^p(\Omega)} = \left(\int_\Omega |u(x)|^p dx \right)^{1/p}. \tag{1}$$

For $p = \infty$, $L^\infty(\Omega)$ is the space of measurable and bounded real functions on Ω. It is also a Banach space for the norm

$$\|u\|_{L^\infty(\Omega)} = \text{ess sup}_{x \in \Omega} |u(x)|. \tag{2}$$

For $p = 2$, $L^2(\Omega)$ is a Hilbert space for the scalar product

$$(u, v) = \int_\Omega u(x) \cdot v(x) dx \tag{3}$$

with the corresponding norm denoted by

$$| \, u \, | = \|u\|_{L^2(\Omega)} = \{(u, u)\}^{1/2}. \tag{4}$$

Let us denote by $W^{s,p}(\Omega)$, $s \in \mathbb{N}, p \in [1, \infty]$, the Sobolev space of functions u in $L^p(\Omega)$ whose distribution derivatives of order less than or equal to s are in $L^p(\Omega)$. It is a Banach space for the norm

$$\|u\|_{W^{s,p}(\Omega)} = \sum_{[\alpha] \leq s} \|D^\alpha u\|_{L^p(\Omega)} \tag{5}$$

where $D_i u = \partial u / \partial x_i$, $i \in [1, n]$, is the notation for the partial differential derivatives of a function u, $D^\alpha u = D_1^{\alpha_1} \ldots D_n^{\alpha_n} u = (\partial^{\alpha_1 + \cdots + \alpha_n} u) / (\partial x_1^{\alpha_1} \ldots \partial x_n^{\alpha_n})$, $\alpha = \{\alpha_1, \ldots, \alpha_n\} \in \mathbb{N}^n$, and $[\alpha] = \alpha_1 + \cdots + \alpha_n$. For $p = 2$, $W^{s,2}(\Omega) = H^s(\Omega)$ and this is a Hilbert space for the scalar product

$$((u, v))_{H^s(\Omega)} = \sum_{[\alpha] \leq s} (D^\alpha u, D^\alpha v). \tag{6}$$

Let us consider the Sobolev spaces $H^s(\Omega)$ which contain $C^\infty(\overline{\Omega})$ and $C^s(\overline{\Omega})$. The closure of $C_0^\infty(\Omega)$ in $H^s(\Omega)$ (resp. $W^{s,p}(\Omega)$) is denoted by $H_0^s(\Omega)$ (resp. $W_0^{s,p}(\Omega)$). In particular,

$$H^1(\Omega) = \{u \in L^2(\Omega), D_i u \in L^2(\Omega), 1 \leq i \leq n\}, \tag{7}$$

$$H_0^1(\Omega) = \text{the closure of } C_0^\infty(\Omega) \text{ in } H^1(\Omega), \tag{8}$$

which are Hilbert spaces for the scalar product

$$((u, v))_{H^1(\Omega)} = (u, v) + \sum_{i=1}^n (D_i u, D_i v). \tag{9}$$

For a bounded Ω, the Poincaré inequality

$$|u| \leq c_0(\Omega)\left\{\sum_{i=1}^{n}|D_i u|^2\right\}^{1/2}, \qquad \forall\, u \in H_0^1(\Omega), \tag{10}$$

implies that $H_0^1(\Omega)$ is a Hilbert space for the scalar product

$$((u,v)) = \sum_{i=1}^{n}(D_i u, D_i v) \tag{11}$$

and that the corresponding norm

$$\|u\| = \{((u,u))\}^{1/2} \tag{12}$$

is equivalent to the norm induced by $H^1(\Omega)$ [17].

3. Reaction-Advection-Diffusion (R-A-D) System

In order to show how to apply the backstepping methodology to the boundary control of a certain class of R-A-D system, we will illustrate the procedure that must be systematically followed to reach the goal.

Let us consider a R-A-D system of the form

$$u_t(x,t) = u_{xx}(x,t) + bu_x(x,t) + \lambda u(x,t), \tag{13}$$

$$u_x(0,t) = -\frac{b}{2}u(0,t), \tag{14}$$

$$u_x(1,t) = \mathcal{U}(t), \tag{15}$$

for a function $u(x,t)$ defined in the spatial variable $x \in [0,1]$ and time $t \in [0,\infty)$, b and λ are arbitrary constant coefficients, $\mathcal{U}(t)$ is the actuation signal (Neumann actuation) and Neumann boundary conditions are imposed. $u_{xx}(x,t)$ is the diffusion term, $bu_x(x,t)$ is the advection term, and $\lambda u(x,t)$ is the reaction term. First of all, let us consider the following change of variables

$$v(x,t) = u(x,t)e^{(b/2)x} \tag{16}$$

in order to eliminate the advection term $u_x(x,t)$. To do this, we calculate the temporal derivative from (16) which results

$$u_t(x,t) = v_t(x,t)e^{-(b/2)x}. \tag{17}$$

Now, calculating the spatial derivatives from (16) these are given by

$$u_x(x,t) = v_x(x,t)e^{-(b/2)x} - \tfrac{b}{2}v(x,t)e^{-(b/2)x}, \tag{18}$$

$$u_{xx}(x,t) = v_{xx}(x,t)e^{-(b/2)x} - bv_x(x,t)e^{-(b/2)x} + \left(\tfrac{b}{2}\right)^2 v(x,t)e^{-(b/2)x}. \tag{19}$$

Then, the R-A-D Equation (13) becomes

$$v_t(x,t) = v_{xx}(x,t) + \lambda_0 v(x,t) \tag{20}$$

with $\lambda_0 = \lambda - \frac{b^2}{4}$. Next, the boundary conditions are calculated by deriving (16), so,

$$v_x(x,t) = u_x(x,t)e^{(b/2)x} + \tfrac{b}{2}u(x,t)e^{(b/2)x} \tag{21}$$

which evaluated at the boundary $x = 0$ it results

$$v_x(0,t) = u_x(0,t) + \frac{b}{2}u(0,t).$$

From (14),

$$\begin{aligned} v_x(0,t) &= -\frac{b}{2}u(0,t) + \frac{b}{2}u(0,t) \\ &= 0. \end{aligned} \tag{22}$$

Now, evaluating (21) for $x = 1$ we get

$$v_x(1,t) = u_x(1,t)e^{b/2} + \frac{b}{2}u(1,t)e^{b/2}. \tag{23}$$

In this way, we have obtained (20) as a reduced model from (13) given in the form of a R-D equation with boundary conditions (22) and (23).

4. Target System

At this point, we must define through a target system the desirable dynamics to be described by (20) in order to that it will behave like a stable system. A R-D system with negative magnitude to the reaction term can be a good choice as an exponentially stable one [5,12]. However, let us consider the heat equation

$$w_t(x,t) = w_{xx}(x,t), \tag{24}$$

$$w_x(0,t) = 0, \tag{25}$$

$$w_x(1,t) = -\frac{1}{2}w(1,t), \tag{26}$$

as target system, as it is usual in boundary control of PDEs [14] due to its physical motivation [4], but with Neumann boundary conditions for a function $w(x,t)$ defined in the spatial variable $x \in [0,1]$ and time $t \in [0,\infty)$. In the following, we must prove that the target system (24)–(26) is exponentially stable in the sense of the L_2-norm of the state $w(x,t)$ with respect to the spatial variable limited to the unit domain. The L_2-norm is defined by

$$\|w(x,t)\| = \left(\int_0^1 w^2(x,t)dx\right)^{1/2}. \tag{27}$$

Let us consider the Lyapunov function

$$V(x,t) = \frac{1}{2}\int_0^1 w^2(x,t)dx, \tag{28}$$

whose time derivative is given by

$$\dot{V}(x,t) = \frac{dV}{dt} = \int_0^1 w(x,t)w_t(x,t)dx. \tag{29}$$

From (24), the time derivative $\dot{V}$ of V can be rewritten as

$$\dot{V}(x,t) = \int_0^1 w(x,t)w_{xx}(x,t)dx. \tag{30}$$

By using integration by parts we obtain

$$\dot{V}(x,t) = w(1,t)w_x(1,t) - w(0,t)w_x(0,t) - \int_0^1 w_x^2(x,t)dx,$$

from which taking into account the boundary conditions (25) and (26) results

$$\dot{V}(x,t) = -\frac{1}{2}w^2(1,t) - \int_0^1 w_x^2(x,t)dx. \tag{31}$$

In order to continue with our analysis, the following lemma which establishes the relationship between the L_2-norms of $w(x,t)$ and $w_x(x,t)$ will be of utility.

Lemma 1 (Poincaré Inequality). *[12] For any $w \in H^1(0,1)$ (Sobolev space) the following inequalities hold*

$$\begin{aligned} \int_0^1 w^2(x,t)dx &\leq 2w^2(1,t) + 4\int_0^1 w_x^2(x,t)dx, \\ \int_0^1 w^2(x,t)dx &\leq 2w^2(0,t) + 4\int_0^1 w_x^2(x,t)dx. \end{aligned} \tag{32}$$

Proof. See [12].
Multiplying both sides of (31) by a constant we get

$$4\dot{V}(x,t) = -2w^2(1,t) - 4\int_0^1 w_x^2(x,t)dx. \tag{33}$$

From Lemma 1, this last identity can be written as

$$\dot{V}(x,t) \leq -\frac{1}{4}\int_0^1 w^2(x,t)dx, \tag{34}$$

which, considering (28), can be rewritten as

$$\dot{V}(x,t) \leq -\frac{1}{2}V(x,t). \tag{35}$$

Let us try to solve this last relation by integration, where (35) can be equivalently written as

$$\frac{d(V(x,t))}{dt} \leq -\frac{1}{2}V(x,t).$$

So, separating variables and considering the integral symbol we get

$$\int \frac{d(V(x,t))}{V(x,t)} \leq -\frac{1}{2}\int dt.$$

Then, solving each integral it yields

$$ln(V(x,t)) \leq -\frac{1}{2}t + c.$$

Since the natural logarithm $ln(x)$ is defined as the inverse function to the exponential function e^x, this last inequality can be expressed by

$$e^{ln(V(x,t))} \leq e^{-(1/2)t}e^c,$$

i.e.,

$$V(x,t) \leq e^{-(1/2)t}e^{c}, \tag{36}$$

from which, setting $t = 0$ results

$$V(x,0) \leq e^{c}. \tag{37}$$

Additionally, we need to prove that $V(x,t) \leq V(x,0)$ in order to guarantee that the Lyapunov function is bounded. So, from the definition for which is desirable that $\dot{V}(x,t)$ be negative semidefinite, i.e.,

$$\dot{V}(x,t) \leq 0, \tag{38}$$

integrating both sides from such inequality, i.e.,

$$\int_0^t \dot{V}(x,t)dt \leq 0 \int_0^t dt,$$

and evaluating limits, it results

$$V(x,t) - V(x,0) \leq 0.$$

Consequently,

$$V(x,t) \leq V(x,0).$$

Thus, from (36)

$$V(x,t) \leq V(x,0)e^{-(1/2)t}, \tag{39}$$

or, by using (28) in terms of the L_2-norm (27),

$$\frac{1}{2}\int_0^1 w^2(x,t)dx \leq e^{-(1/2)t} \cdot \frac{1}{2}\int_0^1 w^2(x,0)dx. \tag{40}$$

Then, taking square root at both sides from (40) it yields

$$\|w(x,t)\| \leq \|w_0(x)\|e^{-(1/4)t} \tag{41}$$

where $w_0(x)$ stands for $w(x,0)$. From the analysis above, we have demonstrated that the PDE system (24)–(26) is exponentially stable in the sense of the L_2-norm of the state $w(x,t)$. □

5. Backstepping Transformation

From the backstepping methodology to the bounded control of PDEs [14], the coordinate transformation, also called the Volterra integral transformation,

$$w(x,t) = v(x,t) - \int_0^x k(x,y)v(y,t)dy \tag{42}$$

is used to transform the system (20), (22) and (23) into the target system (24)–(26). First of all, the requirement about to find a kernel gain $k(x,y)$, that makes that the plant behaves like the target system, must be fulfilled. It is well known that (42) is invertible, so, the smoothness of the kernels of the direct and inverse transformation in x and y establishes the equivalence of norms of v and w in both L^2 and H^1 spaces. Thus, from the properties of the heat Equations (24)–(26) can be concluded that the closed-loop system is exponentially stable in L^2 and H^1.

Let $k(x,t)$ be a continuous function such that its partial derivative is continuous in $x \in [0,1]$ for $t \in [0,\infty)$. From the Leibniz differentiation rule [18]

$$\frac{d}{dx}\int_0^x k(x,y)dy = k(x,x) + \int_0^x k_x(x,y)dy \tag{43}$$

with

$$k_x(x,x) = \frac{\partial}{\partial x}k(x,y)\Big|_{y=x} \tag{44}$$

$$k_y(x,x) = \frac{\partial}{\partial y}k(x,y)\Big|_{y=x} \tag{45}$$

$$\frac{d}{dx}k(x,x) = k_x(x,x) + k_y(x,x), \tag{46}$$

the temporal derivative from (42) is given as

$$w_t(x,t) = v_t(x,t) - \int_0^x k(x,y)v_t(y,t)dy, \tag{47}$$

which, from (20), can be written as

$$w_t(x,t) = v_{xx}(x,t) + \lambda_0 v(x,t) - \int_0^x k(x,y)\left[v_{yy}(y,t) + \lambda_0 v(y,t)\right]dy. \tag{48}$$

From this last equation, applying integration by parts twice for the first term into the integral yields

$$\int_0^x k(x,y)v_{yy}(y,t)dy = k(x,x)v_x(x,t) - k_y(x,x)v(x,t) + k_y(x,0)v(0,t) + \int_0^x k_{yy}(x,y)v(y,t)dy. \tag{49}$$

Then, the spatial derivatives from (42), considering again (43), result

$$w_x(x,t) = v_x(x,t) - k(x,x)v(x,t) - \int_0^x k_x(x,y)v(y,t)dy, \tag{50}$$

$$w_{xx}(x,t) = v_{xx}(x,t) - \tfrac{d}{dx}k(x,x)v(x,t) - k(x,x)v_x(x,t) - k_x(x,x)v(x,t) - \int_0^x k_{xx}(x,y)v(y,t)dy. \tag{51}$$

From (24), subtracting (48) and (49) from (51) yields

$$\begin{aligned} w_t(x,t) - w_{xx}(x,t) = {} & \left(\lambda_0 + k_y(x,x) + \tfrac{d}{dx}k(x,x) + k_x(x,x)\right)v(x,t) \\ & - k_y(x,0)v(0,t) + \int_0^x \left(k_{xx}(x,y) - k_{yy}(x,y) - \lambda_0 k(x,y)\right)v(y,t)dy. \end{aligned} \tag{52}$$

By setting, from the integral term in (52),

$$k_{xx}(x,y) - k_{yy}(x,y) - \lambda_0 k(x,y) = 0 \tag{53}$$

it follows that

$$k_{xx}(x,y) - k_{yy}(x,y) = \lambda_0 k(x,y). \tag{54}$$

Furthermore, from (52), setting equal to zero the term

$$-k_y(x,0)v(0,t) = 0 \tag{55}$$

we have

$$k_y(x,0) = 0. \tag{56}$$

Moreover, from the first term in (52), by setting

$$\lambda_0 + k_y(x,x) + \frac{d}{dx}k(x,x) + k_x(x,x) = 0, \tag{57}$$

from (46), it can be written as

$$\lambda_0 + 2\frac{d}{dx}k(x,x) = 0. \tag{58}$$

Thus, from this last identity, subtracting λ_0 and integrating with the differential dx it yields

$$k(x,x) = -\frac{\lambda_0}{2}x. \tag{59}$$

Hence, in order to zeroing the right-hand side from (52), for all $v(x,t)$, the following identities

$$k_{xx}(x,y) - k_{yy}(x,y) = \lambda_0 k(x,y), \qquad x,y \in [0,1], \tag{60}$$
$$k_y(x,0) = 0, \qquad x \in [0,1], \tag{61}$$
$$k(x,x) = -\frac{\lambda_0}{2}x, \quad x \in [0,1], \tag{62}$$

must be fulfilled.

By inspection of (60) and (61) we have a hyperbolic PDE system whose solution will be the kernel gain $k(x,y)$. We can find the kernel by converting (60) and (61) into an integral equation. To this end, let us define the following change of variables

$$\xi = x + y, \quad \eta = x - y, \tag{63}$$

and let us denoting

$$G(\xi,\eta) = k(x,y). \tag{64}$$

Next, by calculating the derivative with respect to the x and y variables from (64) we have

$$k_x(x,y) = G_\xi(\xi,\eta) + G_\eta(\xi,\eta), \tag{65}$$
$$k_y(x,y) = G_\xi(\xi,\eta) - G_\eta(\xi,\eta), \tag{66}$$
$$k_{xx}(x,y) = G_{\xi\xi}(\xi,\eta) + 2G_{\xi\eta}(\xi,\eta) + G_\eta(\xi,\eta), \tag{67}$$
$$k_{yy}(x,y) = G_{\xi\xi}(\xi,\eta) - 2G_{\xi\eta}(\xi,\eta) + G_\eta(\xi,\eta). \tag{68}$$

Substituting (65)–(68) into (60) it yields

$$4G_{\xi\eta}(\xi,\eta) = \lambda_0 G(\xi,\eta). \tag{69}$$

By applying the change of variables (63) to the condition (61) we get

$$G(\xi,0) = -\frac{\lambda_0}{4}\xi. \tag{70}$$

Then, by applying (70) into (62) it results

$$G_\xi(\eta,\eta) = G_\eta(\eta,\eta). \tag{71}$$

From all of the above, we arrive to the PDE (69) with conditions (70) and (71).

By integrating (69) respect to η, with limits from 0 to η, results

$$\int_0^\eta G_{\eta s}(\xi, s)ds = \frac{\lambda_0}{4}\int_0^\eta G(\xi, s)ds. \tag{72}$$

Evaluating limits from the integral on the left side of (72) it yields

$$G_{\xi}(\xi, \eta) - G_{\xi}(\xi, 0) = \frac{\lambda_0}{4}\int_0^\eta G(\xi, s)ds. \tag{73}$$

From (70), we calculate the derivative respect to ξ to then replace it in (73) yielding

$$G_{\xi}(\xi, \eta) = -\frac{\lambda_0}{4} + \frac{\lambda_0}{4}\int_0^\eta G(\xi, s)ds. \tag{74}$$

Then, by integrating (74) with respect to ξ, for limits from η to ξ, we have

$$\int_\eta^\xi G_\tau(\tau, \eta)d\tau = -\frac{\lambda_0}{4}\int_\eta^\xi d\tau + \frac{\lambda_0}{4}\int_\eta^\xi\int_0^\eta G(\tau, s)dsd\tau. \tag{75}$$

Evaluating limits of both simple integrals in (75) yields

$$G(\xi, \eta) = G(\eta, \eta) - \frac{\lambda_0}{4}(\xi - \eta) + \frac{\lambda_0}{4}\int_\eta^\xi\int_0^\eta G(\tau, s)dsd\tau. \tag{76}$$

Now, we need to express $G(\eta, \eta)$ from this last equation in terms of an integral function. By the identity (71), we can express

$$\frac{d}{d\eta}G(\eta, \eta) = G_\eta(\eta, \eta) + G_{\xi}(\eta, \eta). \tag{77}$$

By applying the identity (46) into (77) results

$$\frac{d}{d\eta}G(\eta, \eta) = G_{\xi}(\eta, \eta) + G_{\xi}(\eta, \eta) \tag{78}$$

$$= 2G_{\xi}(\eta, \eta). \tag{79}$$

By integrating (79) respect to η, with limits from 0 to η, we get

$$\int_0^\eta G_s(s, s)ds = 2\int_0^\eta G_{\xi}(s, s)ds. \tag{80}$$

Evaluating the limits of the integral on the left side from this last equation we write

$$G(\eta, \eta) = G(0, 0) + 2\int_0^\eta G_{\xi}(s, s)ds \tag{81}$$

and, from (70), if $\xi = 0$ then $G(0, 0) = 0$. By the identity (71), if $\xi = \eta$ we can say that both map into the same domain. So, we can express (81) as given by

$$G(\eta, \eta) = 2\int_0^\eta G_{\xi}(\tau, \tau)d\tau. \tag{82}$$

From (74), the integral term is given as

$$G_{\xi}(\tau, \tau) = -\frac{\lambda_0}{4} + \frac{\lambda_0}{4}\int_0^\tau G(\tau, s)ds. \tag{83}$$

So, substituting this last equation in (82) and then expanding it we get

$$G(\eta,\eta) = -\frac{\lambda_0}{2}\int_0^\eta d\tau + \frac{\lambda_0}{2}\int_0^\eta \int_0^\tau G(\tau,s)dsd\tau \tag{84}$$

and evaluating the limits of the first integral on the right side it results

$$G(\eta,\eta) = -\frac{\lambda_0}{2}\eta + \frac{\lambda_0}{2}\int_0^\eta \int_0^\tau G(\tau,s)dsd\tau. \tag{85}$$

By substituting (85) into (76) it yields

$$G(\xi,\eta) = -\frac{\lambda_0}{2}\eta + \frac{\lambda_0}{2}\int_0^\eta \int_0^\tau G(\tau,s)dsd\tau - \frac{\lambda_0}{4}(\xi-\eta) + \frac{\lambda_0}{4}\int_\eta^\xi \int_0^\eta G(\tau,s)dsd\tau, \tag{86}$$

and adding similar terms in (86), this last expression can be written as

$$G(\xi,\eta) = -\frac{\lambda_0}{4}(\xi+\eta) + \frac{\lambda_0}{2}\int_0^\eta \int_0^\tau G(\tau,s)dsd\tau + \frac{\lambda_0}{4}\int_\eta^\xi \int_0^\eta G(\tau,s)dsd\tau. \tag{87}$$

Finally, we have arrived to the integral equation (87) which is equivalent to the PDE (60) with boundary conditions (61) and (62).

6. Integral Equation Solution

Our next goal is to find the solution for (87). By using the method of successive approximations, let us begin with the initial guess

$$G^0(\xi,\eta) = 0, \qquad 0 \le \eta \le \xi \le 2. \tag{88}$$

Here, we have to set a recursive formula for (87) that allow us to approximate the step ahead solution $G^{n+1}(\xi,\eta)$ by using the previous solution with the initial guess as the first solution for $G(\xi,\eta)$. This formula is set up as

$$G^{n+1}(\xi,\eta) = -\frac{\lambda_0}{4}(\xi+\eta) + \frac{\lambda_0}{2}\int_0^\eta \int_0^\tau G^n(\tau,s)dsd\tau + \frac{\lambda_0}{4}\int_\eta^\xi \int_0^\eta G^n(\tau,s)dsd\tau. \tag{89}$$

Let us denote the difference between two consecutive terms as

$$\Delta G^n(\xi,\eta) = G^{n+1}(\xi,\eta) - G^n(\xi,\eta), \tag{90}$$

so,

$$\Delta G^{n+1}(\xi,\eta) = \frac{\lambda_0}{2}\int_0^\eta \int_0^\tau \Delta G^n(\tau,s)dsd\tau + \frac{\lambda_0}{4}\int_\eta^\xi \int_0^\eta \Delta G^n(\tau,s)dsd\tau. \tag{91}$$

Under the assumption that (89) converges to a limit, the solution $G(\xi,\eta)$ can be written as

$$G(\xi,\eta) = \lim_{n\to\infty} G^n(\xi,\eta) \tag{92}$$

or, by using (90), it can be alternatively written as

$$G(\xi,\eta) = \sum_{n=0}^{\infty} \Delta G^n(\xi,\eta). \tag{93}$$

Setting $n = 0$ and taking into account (88)–(91) we calculate

$$\begin{aligned} G^1(\xi,\eta) &= -\frac{\lambda_0}{4}(\xi+\eta), \\ \Delta G^0(\xi,\eta) &= -\frac{\lambda_0}{4}(\xi+\eta), \qquad (94) \\ \Delta G^1(\xi,\eta) &= -\frac{\lambda_0^2}{2(4)}\int_0^\eta\int_0^\tau(\xi+\eta)dsd\tau - \frac{\lambda_0^2}{4^2}\int_\eta^\xi\int_0^\eta(\xi+\eta)dsd\tau, \\ &= -\frac{\lambda_0^2}{2(4^2)}\left[(\xi+\eta)\xi\eta\right]. \qquad (95) \end{aligned}$$

We have already calculated $\Delta G^1(\xi,\eta)$, so using (91) now we get

$$\begin{aligned} \Delta G^2(\xi,\eta) &= -\frac{\lambda_0^3}{(2^2)(4^2)}\int_0^\eta\int_0^\tau[(\xi+\eta)\xi\eta]dsd\tau - \frac{\lambda_0^3}{2(4^3)}\int_\eta^\xi\int_0^\eta[(\xi+\eta)\xi\eta]dsd\tau, \\ &= -\frac{\lambda_0^3}{3(2^2)(4^3)}\left[(\xi+\eta)\xi^2\eta^2\right]. \qquad (96) \end{aligned}$$

At this point, it must be clear that we are obtaining the term (91) for every new value of n. So, for the case when $n = 2$ we only need to calculate $\Delta G^3(\xi,\eta)$, this because $\Delta G^2(\xi,\eta)$ must be calculated for $n = 1$. Hence,

$$\begin{aligned} \Delta G^3(\xi,\eta) &= -\frac{\lambda_0^4}{3(2^3)(4^3)}\int_0^\eta\int_0^\tau[(\xi+\eta)\xi^2\eta^2]dsd\tau - \frac{\lambda_0^4}{3(2^2)(4^4)}\int_\eta^\xi\int_0^\eta[(\xi+\eta)\xi^2\eta^2]dsd\tau, \\ &= -\frac{\lambda_0^4}{(2^2)(3^2)(4^5)}\left[(\xi+\eta)\xi^3\eta^3\right]. \qquad (97) \end{aligned}$$

Now, when setting $n = 3$ it yields

$$\begin{aligned} \Delta G^4(\xi,\eta) &= -\frac{\lambda_0^5}{(2^3)(3^2)(4^5)}\int_0^\eta\int_0^\tau[(\xi+\eta)\xi^3\eta^3]dsd\tau - \frac{\lambda_0^5}{(2^2)(3^2)(4^6)}\int_\eta^\xi\int_0^\eta[(\xi+\eta)\xi^3\eta^3]dsd\tau, \\ &= -\frac{\lambda_0^5}{5(2^2)(3^2)(4^5)}\left[(\xi+\eta)\xi^4\eta^4\right]. \qquad (98) \end{aligned}$$

Thus, we could still evaluating n for any time because from (94)–(98) the pattern to follow is given as

$$\Delta G^n(\xi,\eta) = -\frac{\lambda_0^{n+1}(\xi+\eta)\xi^n\eta^n}{4^{n+1}(n+1)!n!}. \qquad (99)$$

Then, the solution (93) can be expressed as

$$G(\xi,\eta) = -\sum_{n=0}^{\infty}\frac{\lambda_0^{n+1}(\xi+\eta)\xi^n\eta^n}{4^{n+1}(n+1)!n!}. \qquad (100)$$

In order to simplify (100) for software implementation, we use the first-order modified Bessel function of the first kind, i.e.,

$$I_m(x) = \sum_{n=0}^{\infty}\frac{\left(\frac{x}{2}\right)^{m+2n}}{n!(n+m)!}. \qquad (101)$$

By setting $m = 1$, from (101) we have

$$I_1(x) = \sum_{n=0}^{\infty} \frac{\left(\frac{x}{2}\right)^{2n+1}}{n!(n+1)!}. \tag{102}$$

To express (102) in the form (100), rearranging terms in (100) it can be written as

$$G(\xi,\eta) = -\sum_{n=0}^{\infty} \frac{\lambda_0}{4}(\xi+\eta)\left(\frac{\lambda_0\xi\eta}{4}\right)^n \left(\frac{1}{n!(n+1)!}\right). \tag{103}$$

Moreover, separating terms in (102) it can be written as

$$I_1(x) = \sum_{n=0}^{\infty} \frac{x}{2}\left(\frac{x^2}{4}\right)^n \left(\frac{1}{n!(n+1)!}\right). \tag{104}$$

Matching the second term from (103) and (104), i.e.,

$$\left(\frac{x^2}{4}\right)^n = \left(\frac{\lambda_0\xi\eta}{4}\right)^n,$$

it is easy to see that

$$x = \sqrt{\lambda_0\xi\eta}. \tag{105}$$

By the knowledge of (105), the Bessel function can be rewritten as

$$I_1\left(\sqrt{\lambda_0\xi\eta}\right) = \sum_{n=0}^{\infty} \frac{\sqrt{\lambda_0\xi\eta}}{2}\left(\frac{\lambda_0\xi\eta}{4}\right)^n \left(\frac{1}{n!(n+1)!}\right). \tag{106}$$

Now, all the terms appearing in (106) must appear in (100). So, multiplying (103) by

$$\left(\frac{\frac{\sqrt{\lambda_0\xi\eta}}{2}}{\frac{\sqrt{\lambda_0\xi\eta}}{2}}\right) = 1 \tag{107}$$

then it is possible to express (102) in the form (100) as is shown next, i.e.,

$$G(\xi,\eta) = -\frac{\lambda_0}{4}(\xi+\eta)\cdot\sum_{n=0}^{\infty}\left(\frac{\frac{\sqrt{\lambda_0\xi\eta}}{2}}{\frac{\sqrt{\lambda_0\xi\eta}}{2}}\right)\left(\frac{\sqrt{\lambda_0\xi\eta}}{2}\right)^n\left(\frac{1}{n!(n+1)!}\right) \tag{108}$$

$$= -\frac{\lambda_0}{2}(\xi+\eta)\frac{I_1(\sqrt{\lambda_0\xi\eta})}{\sqrt{\lambda_0\xi\eta}}. \tag{109}$$

Taking into account (63) we get

$$\xi + \eta = 2x, \tag{110}$$

$$\sqrt{\lambda_0\xi\eta} = \sqrt{\lambda_0(x^2-y^2)}, \tag{111}$$

so, substituting (110) and (111) into (109) yields

$$k(x,y) = -\lambda_0 x\frac{I_1(\sqrt{\lambda_0(x^2-y^2)})}{\sqrt{\lambda_0(x^2-y^2)}}. \tag{112}$$

So, in the manner described above, the kernel gain (112), key piece in the design of the controller, has been determined. Thus, we have found the solution for $k(x,y)$ in order to achieve the boundary control of the PDE system (20), (22) and (23).

7. Neumann Stabilizing Controller

The control action is applied through the boundary condition (15), i.e., $u_x(1,t)$ is actuated, so by using the first spatial derivative (50) from the Volterra integral transformation (42) and setting $x = 1$ we have

$$w_x(1,t) = v_x(1,t) - k(1,1)v(1,t) - \int_0^1 k_x(1,y)v(y,t)dy. \tag{113}$$

Taking into account (26) and (42) and separating $v_x(1,t)$ from the last equation we get

$$v_x(1,t) = -\frac{1}{2}\left[v(1,t) - \int_0^1 k(1,y)v(y,t)dy\right] + k(1,1)v(1,t) + \int_0^1 k_x(1,y)v(y,t)dy. \tag{114}$$

Considering (62) and from the knowledge of $k(1,1)$, simplifying (114) it can be written as

$$v_x(1,t) = -\left(\frac{\lambda_0+1}{2}\right)v(1,t) + \int_0^1 \left(\frac{k(1,y)}{2} + k_x(1,y)\right)v(y,t)dy. \tag{115}$$

It is easy to get $k(1,y)$ from the kernel gain (112) by setting $x = 1$. In order to get $k_x(1,y)$, then is necessary to know the derivative properties for the Bessel function. The Bessel function must be dependant of a single variable but in (112) it is depending of two variables. Let us consider the next change of variables

$$q(x,y) = \sqrt{\lambda_0}\sqrt{x^2 - y^2} \tag{116}$$

in order to obtain a Bessel function dependant of a single variable. So, we can rewrite (112) as

$$k(q(x,y)) = -\lambda_0 x Q, \tag{117}$$

with $Q = q^{-1}I_1(q)$. The derivative of a Bessel function is given by

$$\frac{d}{dx}(x^{-n}I_n(x)) = x^n I_{n+1}(x). \tag{118}$$

Thus, the derivative of (117) results

$$k_x(q(x,y)) = -\lambda_0\left(Q\frac{d}{dx}(x) + x\frac{d}{dx}(Q)\right).$$

By applying the chain rule to the second term into parentheses it yields

$$k_x(q(x,y)) = -\lambda_0\left(Q\frac{d}{dx}(x) + x\frac{d(Q)}{dq}\frac{dq}{dx}\right), \tag{119}$$

where

$$Q\frac{d}{dx}(x) = q^{-1}I_1(q)(1) \tag{120}$$

and

$$x\frac{d}{dx}(Q)\frac{d}{dx}(q) = x\left(q^{-1}I_2(q)\right)\left(\frac{\sqrt{\lambda_0}x}{\sqrt{x^2-y^2}}\right). \tag{121}$$

Substituting (120) and (121) into (119) results

$$k_x(q(x,y)) = -\lambda_0\left[\frac{I_1(q)}{q} + \frac{x^2\sqrt{\lambda_0}I_2(q)}{q\sqrt{x^2-y^2}}\right]$$

and returning to the original variables x, y we get

$$k_x(x,y) = -\lambda_0\frac{I_1(\sqrt{\lambda_0(x^2-y^2)})}{\sqrt{\lambda_0(x^2-y^2)}} - \lambda_0 x^2\frac{I_2(\sqrt{\lambda_0(x^2-y^2)})}{(x^2-y^2)}. \tag{122}$$

Setting $x = 1$ in (112) and (122), and replacing them in (115) results

$$\begin{aligned} v_x(1,t) = &-\left(\frac{\lambda_0+1}{2}\right)v(1,t) \\ &-\frac{3\lambda_0}{2}\int_0^1\left(\frac{I_1(\sqrt{\lambda_0(1-y^2)})}{\sqrt{\lambda_0(1-y^2)}}\right)v(y,t)dy \\ &-\lambda_0\int_0^1\left(\frac{I_2(\sqrt{\lambda_0(1-y^2)})}{(1-y^2)}\right)v(y,t)dy. \end{aligned} \tag{123}$$

At this point, we need to express the controller in terms of the original variable (16). To this, we set $x = 1$ in (16) and (18) to obtain

$$v(1,t) = u(1,t)e^{(b/2)}, \tag{124}$$

$$u_x(1,t) = v_x(1,t)e^{-(b/2)} - \frac{b}{2}v(1,t)e^{-(b/2)}, \tag{125}$$

respectively. Substituting (123) and (124) into (125) we obtain

$$\begin{aligned} u_x(1,t) = &-\left(\frac{\lambda_0+b+1}{2}\right)u(1,t) \\ &-\frac{3\lambda_0}{2}\int_0^1\frac{I_1(\sqrt{\lambda_0(1-y^2)})}{\sqrt{\lambda_0(1-y^2)}}u(y,t)e^{\frac{b}{2}(y-1)}dy \\ &-\lambda_0\int_0^1\frac{I_2(\sqrt{\lambda_0(1-y^2)})}{(1-y^2)}u(y,t)e^{\frac{b}{2}(y-1)}dy, \end{aligned} \tag{126}$$

which is the Neumann stabilizing controller, i.e., the controller to implement through (15), to the control of the R-A-D system (13)–(15).

8. MATLAB Code

In order to validate the dynamic response for the closed-loop system, in what follows it is described how to implement the whole control system through `MATLAB` [19]. To this end, we use the function `pdepe`. This function is able to solve initial-boundary value problems for parabolic and elliptic PDEs in the one space variable x and time t. This function solves PDEs of the form

$$c\left(x,t,u,\frac{\partial u}{\partial x}\right)\frac{\partial u}{\partial t} = x^{-m}\frac{\partial}{\partial x}\left(x^m f\left(x,t,u,\frac{\partial u}{\partial x}\right)\right) + s\left(s,x,t,\frac{\partial u}{\partial x}\right). \tag{127}$$

By comparing (13) with (127), it can be seen that

$$c\left(x,t,u,\frac{\partial u}{\partial x}\right)=1, \tag{128}$$

$$f\left(x,t,u,\frac{\partial u}{\partial x}\right)=u_x(x,t), \tag{129}$$

$$s\left(s,x,t,\frac{\partial u}{\partial x}\right)=bu_x(x,t)+\lambda u(x,t), \tag{130}$$

$$m=0. \tag{131}$$

From `MATLAB`, the syntax for the `pdepe` function is as follows

```
sol=pdepe(m,pdefun,icfun,bcfun,xlinspace,tlinspace);
```

where `m` is defined in (131), `pdefun` is a function that defines the components of the PDE to be solved, `icfun` is a function that defines the initial condition, `bcfun` is a function that defines the boundary conditions, `xlinspace` is a vector with elements $[x_0, x_1, \ldots, x_n]$ concerning with n specific points (defined by the user) for which a solution is required for every value of `tlinspace`, also a vector with elements $[t_0, t_1, \ldots, t_f]$ for f specific points (also defined by the user).

For all t_0 and x, the solution components are satisfied for all initial conditions of the form

$$u(x,t_0)=u_0(x). \tag{132}$$

Furthermore, for all t and either $x=$ `xlinspace(1)=a` or $x=$ `xlinspace(end)=b`, where `a` represents the left boundary condition and `b` represents the right boundary condition, the solution components must satisfy a boundary condition of the form

$$p(x,t,u)+q(x,t)f\,(x,t,u,)=0. \tag{133}$$

For the PDE (13), based on (128)–(130), `pdefun` function is declared as it is shown below.

```
function[c,f,s]=pdefun(x,t,u,DuDx)
global lambda; global b;
c=1;
f=DuDx;
s=b*f + lambda*u;
```

For the initial condition, `icfun` is declared as follows.

```
function u0=icfun(x)
global initial
global i
u0 = initial(i);
i=i+1;
```

For the boundary conditions (14)–(15), `bcfun` function is declared following the form given by (133). `cntrllr` is for the control law (actuation signal).

```
function[pl,ql,pr,qr] = bcfun(xl,ul,xr,ur,t)
global cntrllr; global b;
pl=b/2*ul;
```

```
ql=1;
pr=-cntrllr;
qr=1;
```

It should be noticed that the functions declared above are needed when running the `pdepe` function. The main code that handles these last functions is shown next.

```
clear all;
global cntrllr; cntrllr=0;
global lambda; lambda=30;
global b; b=10;
global lambda0; lambda0=lambda-b^2/4;
m=0;
```

The spatial vector x is defined below.

```
x0=0; xf=1; xn=40;
x=linspace(x0,xf,xn+1);
```

To define the temporal vector, it must be aware of the next explanation; the solution of a PDE has a mesh where for every spatial value and every temporal value there is a solution value, so we can imagine that for each temporal value there is a spatial vector that we can refer to it as a slice of the solution. We will use two meshes for software implementation: one for a slice of the solution which we call t and the another one for the complete solution of the PDE which we will referring it as T. We need that the slice of the solution has at least three data, so we need to define the step that all slices will have in order to get the solution from the PDE. To this end, the code used is given next.

```
t0=0; step=0.001; tf=step; tn=3;
zf=20; tt=zf*step; ttn=zf*tn;
t=linspace(t0,tf,tn);
T=linspace(t0,tt,ttn+1);
```

`tt` is for the simulation time and `ttn` is for the total number of simulation data, i.e., the sum of all solution slices. The initial conditions are defined through the following code.

```
global initial; initial=5*(1-2*sin(3*pi*x/2));
u=initial;
global i; i=1;
```

`pdepe` function returns the solution in a 3-D array called `sol`, where `sol(i,j,k)` approximates the k-th component of the solution u_k evaluated at $t(i)$ and $x(j)$. The size of `sol` is `ttn`-by-`xn`-by-`initial` and the command to extract the solution of the PDE for the defined size is as follows.

```
u = sol(:,:,1);
```

```
ND=5;
tg=zeros(1,ttn/ND+1);
ug=zeros(ttn/ND+1,xn+1);
ug(1,:)=initial;
```

A `for` cycle is implemented to compute the whole solution of the PDE system. The code shown at the bottom is used to plot the solution.

```
for z=1:zf
sol=pdepe(m,@pdefun,@icfun,@bcfun,x,t);
u=[u; sol(:,:,1)];
root = sqrt(lambda0*(1-x.^2));
bss1 = besseli(1,root);
bss2 = besseli(2,root);
coc1 = (3/2)*bss1./root;
coc2 = bss2./(1-x.^2);
coc1(end) = coc1(end-1)*.89;
coc2(end) = coc2(end-1)*.89;
coc = coc1 + coc2;
int = coc.*exp(b/2*(x-1)).*u(end,:);
integ = trapz(x,int);
u_1 = u(end,end);
cntrllr = -((lambda0+b+1)/2)*u_1 - lambda0*integ;
initial = u(end,:);
i=1;
t0=t0+step;
tf=tf+step;
t=linspace(t0,tf,tn);
end

for h=1:ttn/ND
tg(1,h+1) = T(1,ND*h+1);
ug(h+1,:) = u(ND*h+1,:);
end

figure(1);
surf(x,tg,ug);
view(45,30);
title('Reaction-Advection-Diffusion Equation ','FontSize',12');
xlabel('Distance $x$','Interpreter','Latex','FontSize',12');
ylabel('Time $t$','Interpreter','Latex','FontSize',12');
zlabel('Solution $u(x,t)$','Interpreter','Latex','FontSize',12');
view(55,30)
figure(2);
plot(tg,ug(:,end)');
xlabel('Time $t$','Interpreter','Latex','FontSize',15');
ylabel('Solution $u(1,t)$','Interpreter','Latex','FontSize',15');
grid on;
```

The MATLAB code shown above is contained in four different M-files. The pdefun.m, icfun.m and bcfun.m files correspond to the `pdefun`, `icfun` and `bcfun` functions, respectively. The main code is saved as BCRAD.m which contains the remaining MATLAB code in the order presented above. The BCRAD.m file must be compiled and then executed to get the numerical solution.

9. Simulation Results

In this section, we exhibit simulation results for both open- and closed-loop dynamics of the PDE system (13)–(15). The aim is to show that the Neumann stabilizing controller (126) applied at the boundary condition (15) achieves boundary control of the R-A-D system. The open-loop dynamic behavior is that for which $\mathcal{U}(t) = 0$. The parameters for the PDE (13), and for which the code was executed, are $\lambda = 30$ and $b = 10$ with initial condition $u_0(x) = 5\left(1 - 2\sin(\frac{3\pi x}{2})\right)$. Figure 1 shows the open-loop dynamic behavior of the R-A-D system from which it can be seen that the system is unstable. From Figure 2, it can be seen that the closed-loop response from the boundary control of the PDE system the controller makes of this PDE system a stable one. Figure 3 shows the open-loop response behavior at the right boundary for $u_x(1, t) = 0$. Figure 4 shows the control effort when applying the Neumann stabilizing controller (126) at the right boundary of the R-A-D system in order to implement the closed-loop system. From Figure 4, it can be seen that the control effort is bounded.

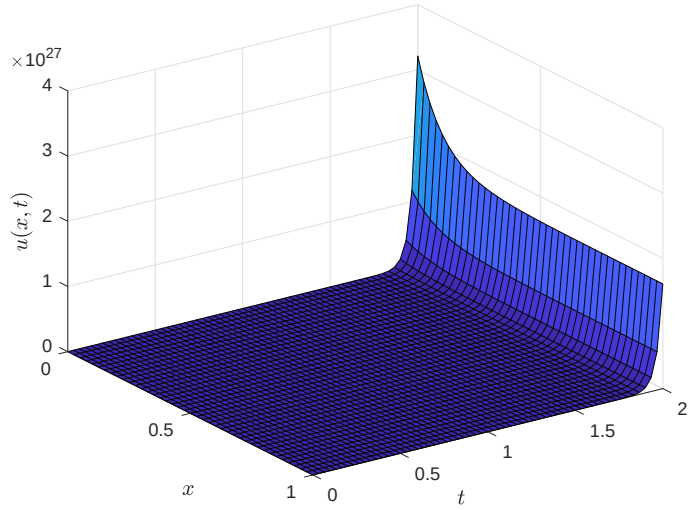

Figure 1. Open-loop response from the solution of the R-A-D PDE.

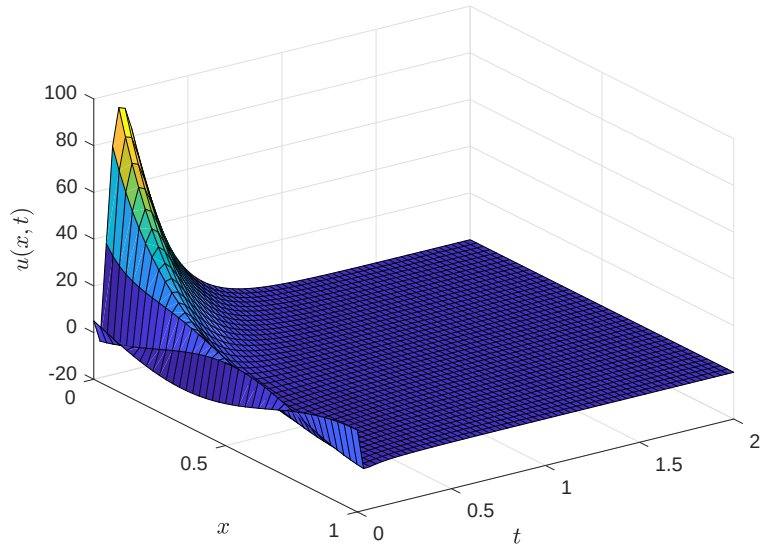

Figure 2. Closed-loop response from the solution to the R-A-D PDE with boundary control.

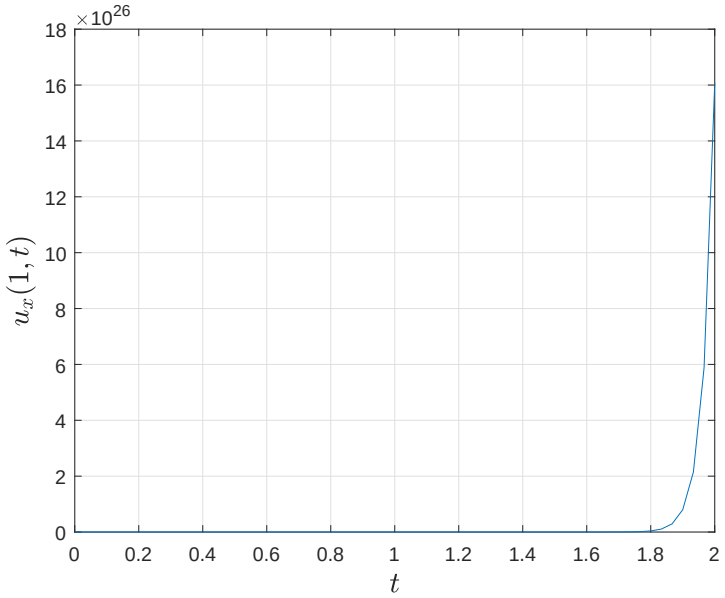

Figure 3. Open-loop behavior at the right hand side boundary from the R-A-D PDE.

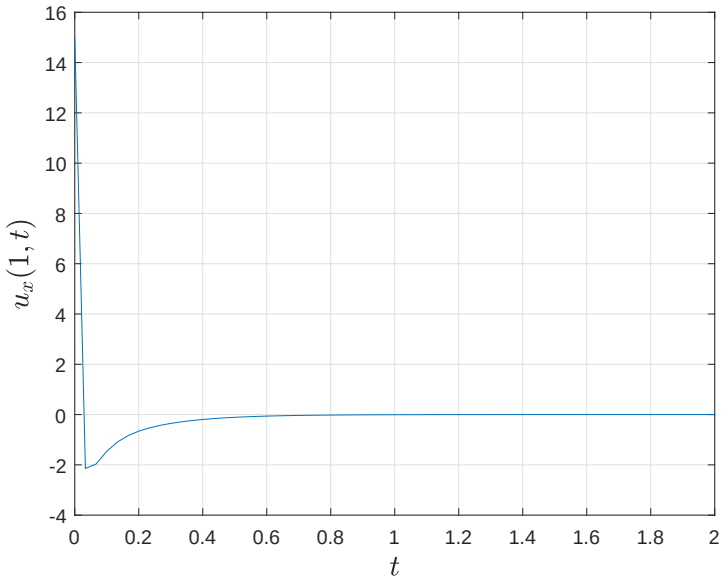

Figure 4. Closed-loop control effort at the right hand side boundary from the R-A-D PDE.

10. Conclusions

In this work, a Neumann stabilizing controller to the boundary control of a certain class of reaction-advection-diffusion (R-A-D) system, with constant parameters to the PDE and Neumann boundary conditions, via backstepping approach is designed. The performance of the stabilizing controller is validated via numerical simulation. Simulation results show that the control goal has been achieved. Furthermore, our numerical algorithm based on MATLAB® is provided in order to replicate the simulation results. It should be noticed that our numerical algorithm also has been validated for others problems about boundary control of PDEs but with the restriction that the PDEs must be of parabolic or elliptic type. As future work, from the R-A-D system used here, the approaches proposed in [20–23] could be explored.

Author Contributions: Conceptualization, E.C.-Q. and F.J.; methodology, E.C.-Q. and F.J.; software, E.C.-Q. and F.J.; validation, F.J.; formal analysis, E.C.-Q. and F.J.; investigation, E.C.-Q. and F.J.; data curation, E.C.-Q. and F.J.; visualization, E.C.-Q. and F.J.; writing–original draft preparation, E.C.-Q. and F.J.; writing–review and editing, F.J.; supervision, F.J.; resources, F.J.; funding acquisition, F.J.; project administration, F.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research work was funded by Tecnológico Nacional de México (TecNM) grants.

Acknowledgments: This work was supported by CONACYT México, through grant 436077, and by TecNM Projects.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DPSs	Distributed Parameters Systems
ISS	Input-to-State Stability
ODEs	Ordinary Differential Equations
PDEs	Partial Differential Equations
P(I)DEs	Partial Integro-Differential Equations
R-A-D	Reaction-Advection-Diffusion
R-D	Reaction-Diffusion

References

1. Haberman, R. *Elementary Applied Partial Differential Equations: With Fourier Series and Boundary Value Problems*; Prentice Hall: Englewood Cliffs, NJ, USA, 1987.
2. Krstic, M.; Kanellakopoulos, I.; Petar, V. *Nonlinear and Adaptive Control Design*; Wiley: New York, NY, USA, 1995.
3. Khalil, H. *Nonlinear Systems*; Prentice Hall: Englewood Cliffs, NJ, USA, 2002.
4. Balogh, A.; Miroslav, K. Infinite Dimensional Backstepping-Style Feedback Transformations for a Heat Equation with an Arbitrary Level of Instability. *Eur. J. Control* **2002**, *8*, 165–175.
5. Liu, W. Boundary Feedbak Stabilization of an Unstable Heat Equation. *SIAM J. Control Optim.* **2003**, *42*, 1033–1043.
6. Smyshlyaev, A.; Krstic, M. Closed-Form Boundary State Feedbacks for a Class of 1-D Partial Integro-Differential Equations. *IEEE Trans. Autom. Control* **2004**, *49*, 2185–2202.
7. Smyshlyaev, A.; Krstic, M. Backstepping Observers for a Class of Parabolic PDEs. *Syst. Control Lett.* **2005**, *54*, 613–625.
8. Smyshlyaev, A.; Krstic, M. Output-Feedback Adaptive Control for Parabolic PDEs with Spatially Varying Coefficients. In Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, CA, USA, 13–15 December 2006.
9. Smyshlyaev, A.; Krstic, M. Adaptive Boundary Control for Unstable Parabolic PDEs—Part II: Estimation-based Designs. *Automatica* **2007**, *43*, 1543–1556.
10. Smyshlyaev, A.; Krstic, M. Adaptive Boundary Control for Unstable Parabolic PDEs—Part III: Output Feedback Examples with Swapping Identifiers. *Automatica* **2007**, *43*, 1557–1564.
11. Krstic, M.; Smyshlyaev, A. Adaptive Boundary Control for Unstable Parabolic PDEs—Part I: Lyapunov Design. *IEEE Trans. Autom. Control* **2008**, *53*, 1575–1591.
12. Smyshlyaev, A.; Krstic, M. *Adaptive Control of Parabolic PDEs*; Princeton University Press: Princeton, NJ, USA, 2010.
13. Krstic, M.; Smyshlyaev, A. *Boundary Control of PDEs: The Backstepping Approach in The Control Handbook,* 2nd ed.; Levine, W.S., Ed.; CRC Press: Boca Raton, FL, USA, 2011.
14. Krstic, M.; Smyshlyaev, A. *Boundary Control for PDEs: A Course on Backstepping Designs*; SIAM: Philadelphia, PA, USA, 2008.
15. Anfinsen, H.; Aamo, O.M. *Adaptive Control of Hyperbolic PDEs*; Springer: Cham, Switzerland, 2019.
16. Karafyllis, I.; Krstic, M. *Input-to-State Stability for PDEs*; Springer: Cham, Switzerland, 2019.

17. Temam, R. *Infinite-Dimensional Dynamical Systems in Mechanics and Physics*; Springer: New York, NY, USA, 1998.
18. Abramowitz, M.; Stegun, I.A. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*; Dover Publications: Mineola, NY, USA, 1965.
19. *MATLAB R2018a*; ©1984-2018; The MathWorks, Inc.: Natick, MA, USA, 2018.
20. Vazquez, R.; Krstic, M. Boundary Control of Coupled Reaction-Advection-Diffusion Systems with Spatially-Varying Coefficients. *IEEE Trans. Autom. Control* **2016** *62*, 2026–2033.
21. Pisano, A.; Baccoli, A.; Orlov, Y.; Usai, E. Boundary control of coupled reaction-advection-diffusion equations having the same diffusivity parameter. *IFAC-PapersOnLine* **2016**, *49*, 86–91.
22. Ren, Z.; Xu, C.; Zhou, Z.; Wu, Z.; Chen, T. Boundary stabilization of a class of reaction–advection–diffusion systems via a gradient-based optimization approach. *J. Frankl. Inst.* **2019**, *356*, 173–195.
23. Wang, S.; Diagne, M.; Qi, J. Adaptive Control of Reaction-Advection-Diffusion PDEs with Distributed Actuation and Unknown Input Delay. In Proceedings of the American Control Conference (ACC), Denver, CO, USA, 1–3 July 2020.