Chapter 4
Parabolic PDEs: Reaction-Advection-Diffusion and Other Equations

This is the most important chapter of this book. In this chapter we present the first designs of feedback laws for stabilization of PDEs using boundary control and introduce the method of backstepping.

As the reader shall learn throughout this book, there are many classes of PDEs—first and second order in time; first, second, third, fourth (and so on) order in space;\(^1\) systems of coupled PDEs of various classes; PDEs interconnected with ODEs; real-valued and complex-valued PDEs;\(^2\) and various other classes. Our introduction to boundary control, stabilization of PDEs, and the backstepping method is presented in this chapter for parabolic PDEs. There is no strong pedagogical reason why the introduction could not be done on some of the other classes of PDEs; however, parabolic PDEs are particularly convenient because they are both sufficiently simple and sufficiently general to serve as a launch pad from which one can easily extend the basic design tools to other classes of PDEs.

Parabolic PDEs are first order in time, which makes them more easily accessible to a reader with a background in ODEs, as opposed to second order in time PDEs such as wave equations, which, as we shall see in Chapter 7, have peculiarities that make defining the system “state,” choosing a Lyapunov function, and adding damping to the system rather nonobvious.

This book deals exclusively with boundary control of PDEs. In-domain actuation of any kind (point actuation or distributed actuation) is not dealt with. The reasons for this are twofold. First, a considerable majority of problems in PDE control, particularly those involving fluids, can be actuated in a physically reasonable way only from the boundary. Second, the backstepping approach is particularly well suited for boundary control. Its earlier ODE applications provide a clue that it should be applicable also to many problems with in-domain actuation; however, at the moment, backstepping for PDEs is developed only for boundary control actuation.

\(^1\) Respectively, we mean the transport equation, the heat and wave equations, the Kurttewege–de Vries equation, and the Euler–Bernoulli beam and Karamoto–Sivashinsky equations.

\(^2\) Respectively, we mean the heat equation, the Schrödinger equation.

4.1 Backstepping: The Main Idea

Let us start with the simplest unstable PDE, the reaction-diffusion equation:

\[
\begin{align*}
u_t(x, t) &= \nu_{xx}(x, t) + \lambda \nu(x, t), \quad (4.1) \\
v(0, t) &= 0, \quad (4.2) \\
v(1, t) &= U(t), \quad (4.3)
\end{align*}
\]

where \(\lambda\) is an arbitrary constant and \(U(t)\) is the control input. The open-loop system (4.1), (4.2) (with \(u(1, t) = 0\)) is unstable with arbitrarily many unstable eigenvalues for sufficiently large \(\lambda\).

Since the term \(\lambda \nu\) is the source of instability, the natural objective for a boundary feedback is to “eliminate” this term. The main idea of the backstepping method is to use the coordinate transformation

\[
w(x, t) = u(x, t) - \int_0^x k(x, y)u(y, t) \, dy \quad (4.4)
\]

along with feedback control

\[
u(1, t) = \int_0^1 k(1, y)u(y, t) \, dy \quad (4.5)
\]

to transform the system (4.1), (4.2) into the target system

\[
\begin{align*}
w_t(x, t) &= w_{xx}(x, t), \quad (4.6) \\
w(0, t) &= 0, \quad (4.7) \\
w(1, t) &= 0 \quad (4.8)
\end{align*}
\]

which is exponentially stable, as shown in Chapter 2. Note that the boundary conditions (4.2), (4.7) and (4.5), (4.8) are verified by (4.4) without any condition on \(k(x, y)\).
4.2. Gain Kernel PDE

The transformation (4.4) is called Volterra integral transformation. Its most characteristic feature is that the limits of integral range from 0 to x, not from 0 to 1. This makes it "spatially causal"; that is, for a given x the right-hand side of (4.4) depends only on the values of u in the interval [0, x]. Another important property of the Volterra transformation is that it is invertible, so that stability of the target system translates into stability of the closed-loop system consisting of the plant plus boundary feedback (see Section 4.5).

Our goal now is to find the function k(x, y) (which we call the "gain kernel") that makes the plant (4.1), (4.2) with the controller (4.5) behave as the target system (4.6)–(4.8). It is not obvious at this point that such a function even exists.

4.2 Gain Kernel PDE

To find out what conditions k(x, y) has to satisfy, we simply substitute the transformation (4.4) into the target system (4.6)–(4.8) and use the plant equations (4.1), (4.2). To do that, we need to differentiate the transformation (4.4) with respect to x and t, which is easy once we recall the Leibniz differentiation rule:

\[
\frac{d}{dx} \int_0^x f(x, y) \, dy = f(x, x) + \int_0^x f_x(x, y) \, dy.
\]

We also introduce the following notation:

\[
k_x(x, x) = \frac{\partial}{\partial x} k(x, x)|_{x=x},
\]
\[
k_y(x, x) = \frac{\partial}{\partial y} k(x, x)|_{x=x},
\]
\[
\frac{d}{dx} k(x, x) = k_x(x, x) + k_y(x, x).
\]

Differentiating the transformation (4.4) with respect to x gives

\[
w_x(x) = u_x(x) - k(x, x)u(x) - \int_0^x k_y(x, y)u(y) \, dy,
\]
\[
w_{xx}(x) = u_{xx}(x) - \frac{d}{dx} (k(x, x)u(x)) - k_x(x, x)u(x) - \int_0^x k_{xy}(x, y)u(y) \, dy.
\]

\[
= u_{xx}(x) - u(x) \frac{d}{dx} k(x, x) - k(x, x)u(x) - k_x(x, x)u(x)
\]
\[
- \int_0^x k_{xy}(x, y)u(y) \, dy.
\]

This expression for the second spatial derivative of \(w(x)\) is going to be the same for different problems since no information about the specific plant and target system is used at this point.

3.2 Chapter 4. Parabolic PDEs: Reaction-Advection-Diffusion and Other Equations

Next, we differentiate the transformation (4.4) with respect to time:

\[
w_t(x) = u_t(x) - \int_0^x k(x, y)u_t(y) \, dy
\]
\[
= u_{xt}(x) + \lambda u_t(x) - \int_0^x k(x, y) (u_{yt}(y) + \lambda u_t(y)) \, dy
\]
\[
= u_{xx}(x) + \lambda u_t(x) - k(x, x)u_t(x) + k(x, 0)u_t(0)
\]
\[
+ \int_0^x k_x(x, y)u_t(y) \, dy - \int_0^x \lambda k(x, y)u(y) \, dy \text{ (integration by parts)}
\]
\[
= u_{xx}(x) + \lambda u_t(x) - k(x, x)u_t(x) + k(x, 0)u_t(0) + k_x(x, x)u(x) - k_x(x, 0)u(0)
\]
\[
- \int_0^x k_{xx}(x, y)u(y) \, dy - \int_0^x \lambda k(x, y)u(y) \, dy \text{ (integration by parts)}.
\]

Subtracting (4.9) from (4.10), we get

\[
w_t - w_{xx} = \left[ \lambda + \frac{d}{dx} k(x, x) \right] u(x) + k(x, 0)u_t(0)
\]
\[
+ \int_0^x \left( k_{xx}(x, y) - k_{xy}(x, y) - \lambda k(x, y) \right) u(y) \, dy.
\]

For the right-hand side to be zero for all u, the following three conditions have to be satisfied:

\[
k_{xx}(x, y) - k_{xy}(x, y) - \lambda k(x, y) = 0,
\]
\[
k(x, 0) = 0,
\]
\[
\lambda + \frac{d}{dx} k(x, x) = 0.
\]

We can simplify (4.14) by integrating it with respect to x and noting from (4.13) that \(k(0, 0) = 0\), which gives us the following:

\[
k_{xx}(x, y) - k_{xy}(x, y) = \lambda k(x, y),
\]
\[
k(x, 0) = 0,
\]
\[
k(x, x) = \frac{\lambda}{2} x.
\]

It turns out that these three conditions are compatible and in fact form a well-posed PDE. This PDE is of hyperbolic type: one can think of it as a wave equation with an extra term \(\lambda k\) (\(k\) plays the role of time). In quantum physics such PDEs are called Klein–Gordon PDEs. The domain of this PDE is a triangle \(0 \leq y \leq x \leq 1\) and is shown in Figure 4.1. The boundary conditions are prescribed on two sides of the triangle and the third side (after solving for \(k(x, y)\) gives us the control gain \(k(1, y)\)).

In the next two sections we prove that the PDE (4.15) has a unique twice continuously differentiable solution.
4.3 Converting the Gain Kernel PDE into an Integral Equation

To find a solution of the PDE (4.15) we first convert it into an integral equation. Introducing the change of variables

\[ \xi = x + y, \quad \eta = x - y, \]  

we have

\[
\begin{align*}
    k(x, y) &= G(\xi, \eta), \\
    k_x &= G_x + G_y, \\
    k_{xx} &= G_{xx} + 2G_x + G_{yy}, \\
    k_y &= G_x - G_y, \\
    k_{yy} &= G_{yy} + 2G_y + G_{xy}.
\end{align*}
\]

Thus, the gain kernel PDE becomes

\[
\begin{align*}
    G_{\xi\eta}(\xi, \eta) &= \frac{\lambda}{4} G(\xi, \eta), \\
    G(\xi, 0) &= 0, \\
    G(\xi, 0) &= -\frac{\lambda}{4} \xi.
\end{align*}
\]

Integrating (4.17) with respect to \( \eta \) from 0 to \( \eta \), we get

\[
G(\xi, \eta) = G(\xi, 0) + \int_0^\eta \frac{\lambda}{4} G(\xi, s) \, ds = -\frac{\lambda}{4} \xi + \frac{\lambda}{4} \int_0^\eta G(\xi, s) \, ds.
\]

Next, we integrate (4.20) with respect to \( \xi \) from \( \eta \) to \( \xi \) to get

\[
G(\xi, \eta) = G(\xi, \eta) - \frac{\lambda}{4} (\xi - \eta) + \frac{\lambda}{4} \int_{\eta}^{\xi} G(\xi, s) \, ds.
\]

We obtain the integral equation, which is equivalent to PDE (4.15) in the sense that every solution of (4.15) is a solution of (4.21). The point of converting the PDE into the integral equation is that the latter is easier to analyze with a special tool, which consider next.

4.4 Method of Successive Approximations

The method of successive approximations is conceptually simple: start with an initial guess for a solution of the integral equation, substitute it into the right-hand side of the equation, then use the obtained expression as the next guess in the integral equation and repeat the process. Eventually this process results in a solution of the integral equation.

Let us start with an initial guess

\[
G^0(\xi, \eta) = 0
\]

and set up the recursive formula for (4.21) as follows:

\[
G^{n+1}(\xi, \eta) = -\frac{\lambda}{4} (\xi - \eta) + \frac{\lambda}{4} \int_0^\xi \int_0^\xi G^n(\tau, \tau') \, d\tau \, d\tau'.
\]

If this converges, we can write the solution \( G(\xi, \eta) \) as

\[
G(\xi, \eta) = \lim_{n \to \infty} G^n(\xi, \eta).
\]

Let us denote the difference between two consecutive terms as

\[
\Delta G^n(\xi, \eta) = G^{n+1}(\xi, \eta) - G^n(\xi, \eta).
\]

Then

\[
\Delta G^{n+1}(\xi, \eta) = \frac{\lambda}{4} \int_0^\xi \int_0^\xi \Delta G^n(\tau, \tau') \, d\tau \, d\tau'.
\]

and (4.24) can be alternatively written as

\[
G(\xi, \eta) = \sum_{n=0}^{\infty} \Delta G^n(\xi, \eta).
\]

Computing \( \Delta G^n \) from (4.26) starting with

\[
\Delta G^0 = G^1(\xi, \eta) = -\frac{\lambda}{4} (\xi - \eta),
\]

we can observe the pattern which leads to the following formula:

\[
\Delta G^n(\xi, \eta) = \frac{(\xi - \eta) n \eta^n}{n!(n + 1)!} \left( \frac{\lambda}{4} \right)^{n+1}.
\]

This formula can be verified by induction. The solution to the integral equation is given by

\[
G(\xi, \eta) = \sum_{n=0}^{\infty} \frac{(\xi - \eta) n \eta^n}{n!(n + 1)!} \left( \frac{\lambda}{4} \right)^{n+1}.
\]
4.5 Inverse Transformation

To compute the series (4.30), note from the appendix that a first-order modified Bessel function of the first kind can be represented as

\[ I_1(x) = \sum_{n=0}^{\infty} \frac{(\frac{x}{2})^{2n+1}}{n!(n+1)!}. \]  

Comparing this expression with (4.30), we obtain

\[ G(\xi, \eta) = -\frac{\lambda}{2} (\xi - \eta) I_1(\sqrt{\lambda(\xi^2 - \eta^2)}) \sqrt{\lambda(\xi^2 - \eta^2)}. \]  

or, returning to the original \( x, y \) variables,

\[ k(x, y) = -\lambda y I_1\left(\sqrt{\lambda(x^2 - y^2)}\right). \]  

In Figure 4.2 the control gain \( k(1, y) \) is shown for different values of \( \lambda \). Obviously, as \( \lambda \) gets larger, the plant becomes more unstable, which requires more control effort. Low control gain near the boundaries is also logical: near \( x = 0 \) the state is small even without control because of the boundary condition \( u(0) = 0 \), and near \( x = 1 \) the control has the most authority.

4.5 Inverse Transformation

To complete the design we need to establish that stability of the target system (4.6)–(4.8) implies stability of the closed-loop plant (4.1), (4.2), (4.5). In other words, we need to show that the transformation (4.4) is invertible.

36 Chapter 4. Parabolic PDEs: Reaction-Advection-Diffusion and Other Equations

Let us write an inverse transformation in the form

\[ u(x) = w(x) + \int_0^x l(x, y) w(y) \, dy, \tag{4.34} \]

where \( l(x, y) \) is the transformation kernel.

Given the direct transformation (4.4) and the inverse transformation (4.34), the kernels \( k(x, y) \) and \( l(x, y) \) satisfy

\[ l(x, y) = k(x, y) + \int_y^x k(x, \xi) l(\xi, y) \, d\xi. \]  

Proof of (4.35). First, let us recall from calculus the following formula for changing the order of integration:

\[ \int_0^x \int_0^x f(x, y, \xi) \, d\xi \, dy = \int_0^x \int_0^x f(\xi, y, x) \, dy \, d\xi. \]  

Substituting (4.34) into (4.4), we get

\[ u(x) = w(x) + \int_0^x l(x, y) w(y) \, dy - \int_0^x k(x, y) w(y) \, dy + \int_0^x k(x, y) l(x, y) w(y) \, dy \, d\xi. \]

\[ = w(x) + \int_0^x l(x, y) w(y) \, dy - \int_0^x k(x, y) w(y) \, dy - \int_0^x k(x, y) l(x, y) w(y) \, d\xi. \]

Since the last line has to hold for all \( w(y) \), we get the relationship (4.35). \( \square \)

The formula (4.35) is general (it does not depend on the plant and the target system) but is not very helpful in actually finding \( l(x, y) \) from \( k(x, y) \). Instead, we follow the same approach that led us to the kernel PDE for \( k(x, y) \): we differentiate the inverse transformation (4.34) with respect to \( x \) and \( t \) and use the plant and the target system to obtain the PDE for \( l(x, y) \).

Differentiating (4.34) with respect to \( x \), we get

\[ u_x(x) = w_x(x) + \int_0^x l_x(x, y) w(y) \, dy \]

\[ = w_x(x) + l(x, x) w_x(x) - l(x, 0) w_x(0) - l_x(x, x) w(x) \]

\[ + \int_0^x l(x, y) w(y) \, dy, \tag{4.37} \]

and differentiating twice with respect to \( x \) gives

\[ u_{xx}(x) = w_{xx}(x) + l_{xx}(x, x) w(x) + w(x) \frac{d}{dx} l(x, x) + l(x, x) w_x(x) \]

\[ + \int_0^x l_{xx}(x, y) w(y) \, dy. \tag{4.38} \]
4.5. Inverse Transformation

Subtracting (4.38) from (4.37), we get
\[
\lambda w(x) + \int_0^t l(x, y) u(y) dy = -2w(x) \frac{d}{dx}l(x, x) - l(x, 0) w_t(0)
\]
\[
+ \int_0^t (l_x(x, y) - \lambda l_t(x, y)) w(y) dy,
\]
which gives the following conditions on \( l(x, y) \):
\[
l_{xx}(x, y) - l_{yy}(x, y) = -\lambda l(x, y),
\]
\( l(x, 0) = 0, \) \hspace{1cm} (4.39)
\( l(x, x) = \frac{\lambda}{2}. \) \hspace{1cm} (4.40)
\( l(x, x) = \frac{\lambda}{2}. \)
\hspace{1cm} (4.41)

Comparing this PDE with the PDE (4.15) for \( k(x, y) \), we see that
\[
l(x, y; \lambda) = -k(x, y; \lambda).
\]
\hspace{1cm} (4.42)

From (4.33) we have
\[
l(x, y) = -\lambda y \frac{f_1 \left( \sqrt{\lambda (x^2 - y^2)} \right)}{f_1 \left( \sqrt{\lambda (x^2 - y^2)} \right)} - \lambda y \frac{f_1 \left( \sqrt{\lambda (x^2 - y^2)} \right)}{f_1 \left( \sqrt{\lambda (x^2 - y^2)} \right)}
\]
or, using the properties of \( f_1 \) (see the appendix),
\[
l(x, y) = -\lambda y \frac{f_1 \left( \sqrt{\lambda (x^2 - y^2)} \right)}{f_1 \left( \sqrt{\lambda (x^2 - y^2)} \right)}.
\]
\hspace{1cm} (4.43)

A summary of the control design for the plant (4.13) in Chapter 4 is presented in Table 4.1.

Example 4.1 Consider the plant
\[
u_t = w_{xx} + \lambda w,
\]
\( u(0) = 0, \) \hspace{1cm} (4.52)
\( w(1) = 0, \) \hspace{1cm} (4.53)
\( w(1) = 0, \)
\hspace{1cm} (4.54)

We use the transformation
\[
w(x) = u(x) - \int_0^x k(x, y) u(y) dy
\]
\hspace{1cm} (4.55)
to map this plant into the target system
\[
w_t = w_{yy},
\]
\( w_t(0) = 0, \) \hspace{1cm} (4.56)
\( w(1) = 0, \) \hspace{1cm} (4.57)
\( w(1) = 0, \)
4.5. Inverse Transformation

Figure 4.3. Simulation results for reaction-diffusion plant (4.44), (4.45). Top: open-loop response; Bottom: closed-loop response with controller (4.46) implemented.

Differentiation of the transformation (4.55) with respect to \( x \) gives (4.9) (which does not depend on the particular plant). Differentiating (4.55) with respect to time, we get

\[
w_t(x) = u_t(x) - \int_0^x k(x, y)u_t(y) \, dy
\]

\[
w_t(x) = u_t(x) + \lambda u(x) - \int_0^x k(x, y)u_t(y) \, dy
\]

\[
w_t(x) = u_t(x) + \lambda u(x) - k(x, x)u_t(x) + k(x, 0)u_t(0)
\]

\[
+ \int_0^x k_t(x, y)u(y) \, dy - \int_0^x \lambda k(x, y)u(y) \, dy \quad \text{(integration by parts)}
\]

\[
w_t(x) = u_t(x) + \lambda u(x) - k(x, x)u_t(x) + k(x, 0)u_t(0)
\]

\[
- \int_0^x k_t(x, y)u(y) \, dy + \int_0^x \lambda k(x, y)u(y) \, dy \quad \text{(integration by parts)}.
\]

Figure 4.4. The control (4.46) for reaction-diffusion plant (4.44), (4.45).

Subtracting (4.9) from (4.59), we get

\[
w_t - w_{xx} = \left[ \lambda + 2 \frac{d}{dx} k(x, x) \right] u(x) - k(x, 0)u(0)
\]

\[
+ \int_0^x \left( k_t(x, y) - k_{xx}(x, y) - \lambda k(x, y) \right) u(y) \, dy.
\]

(4.60)

For the right-hand side of this equation to be zero for all \( u(x) \), the following three conditions must be satisfied:

\[
k_t(x, y) - k_{xx}(x, y) - \lambda k(x, y) = 0,
\]

\[
k_t(x, 0) = 0,
\]

\[
\lambda + 2 \frac{d}{dx} k(x, x) = 0.
\]

(4.61)

(4.62)

(4.63)

Integrating (4.63) with respect to \( x \) gives \( k_t(x, x) = -\lambda/2x + k(0, 0) \), where \( k(0, 0) \) is obtained using the boundary condition (4.57),

\[
w_t(x) = u_t(x) + k(x, 0)u(0) = 0,
\]

so that \( k(0, 0) = 0 \). The gain kernel PDE is thus

\[
k_t(x, y) - k_{xx}(x, y) = \lambda k(x, y),
\]

\[
k_t(x, 0) = 0,
\]

\[
k(x, x) = -\frac{\lambda}{2x}.
\]

(4.64)

(4.65)

(4.66)

Note that this PDE is very similar to (4.15); the only difference is in the boundary condition at \( y = 0 \). The solution to the PDE (4.64)-(4.66) is obtained through a summation of