Contents

Chapter 2
Algorithms as Dynamical Systems with Feedback

Algorithms are inventions which very often appear to have little or nothing in common with one another. As a result, it was held for a long time that a coherent theory of algorithms could not be constructed. The last few years have shown that this belief was incorrect, that most convergent algorithms share certain basic properties, and hence a unified approach to algorithms is possible.

—E. L. Polak [Pol71]

At the risk of oversimplification, it can be said that the design of a successful numerical algorithm usually involves the choice of some parameters in such a way that a suitable measure of some residue or error decreases to a reasonably small value as fast as possible. Although this is the case with most numerical algorithms, they are usually analyzed on a case by case basis: there is no general framework to guide the beginner, or even the expert, in the choice of these parameters. At a more fundamental level, one can even say that the very choice of strategy that results in the introduction of the parameters to be chosen is not usually discussed. Thus, the intention of this chapter, and of this book, is to revisit the question raised in the above quote, suggesting that control theory provides a framework for the design or discovery of algorithms in a systematic way.

Control theory, once again oversimplifying considerably, is concerned with the problem of *regulation*. Given a system model, generally referred to as a *plant*, that describes the behavior of some variables to be controlled, the problem of regulation is that of finding a mechanism that either keeps or regulates these variables at constant values, despite changes or disturbances that may act on the system as a whole. A fundamental idea is that of *feedback*: The variable to be controlled is compared with the constant value that is desired, and a difference (error, or residue) variable is generated. This error variable is used (*fed back*) by a parameter-dependent control mechanism to influence the plant in such a way that the controlled variable is driven (or converges) to the desired value. This results in zero error and, consequently, zero control action, as long as no disturbance occurs. The point to be emphasized here is that, in the six decades or so of development of mathematical control theory, several approaches have been developed to the systematic introduction and choice of the so-called feedback control parameters in the regulation problem. One objective of this chapter is to show that one of these approaches—the control Liapunov function approach—

can be used, in a simple and systematic manner, to motivate and derive several iterative methods, both standard as well as new ones, by viewing them as dynamical systems with feedback control.

2.1 Continuous-Time Dynamical Systems that Find Zeros

We start out with a discussion of how one might arrive at a continuous-time dynamical system that finds a simple zero of a given nonlinear vector function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ from a feedback control perspective. In other words, the problem is to find a vector $\mathbf{x}^* \in \mathbb{R}^n$ such that

$$\mathbf{f}(\mathbf{x}^*) = \mathbf{0}. \tag{2.1}$$

For a general nonlinear function $\mathbf{f}(\cdot)$, several solutions will, in general, exist. For the moment, we will content ourselves with finding a simple (i.e., nonrepeated) zero. Let $\mathbf{x}, \mathbf{r} \in \mathbb{R}^n$ such that

$$\mathbf{r} = -\mathbf{f}(\mathbf{x}). \tag{2.2}$$

The variable $\mathbf{r}$ is, in fact, the familiar *residue* of numerical analysis, since its norm can be interpreted as a measure of how far the current guess $\mathbf{x}$ is from a zero of $\mathbf{f}(\cdot)$, i.e., $\mathbf{r} := \mathbf{0} - \mathbf{f}(\mathbf{x})$. The other names that it goes by are *error* and *deviation*. Note that if $\mathbf{f} = \mathbf{A}\mathbf{x} - \mathbf{b}$, then zeroing the residue $\mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}$ corresponds to solving the classical linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$.

In order to introduce control concepts, the first step is to observe that, if the residue is thought of as a time-dependent variable $\mathbf{r}(t)$ that is to be driven to zero, then the variable $\mathbf{x}(t)$ is correspondingly driven to a solution of (2.1). The second step is to assume that this will be done using a suitably defined control variable $\mathbf{u}(t)$, acting directly on the variable $\mathbf{x}(t)$. In control terms, this is written as the following simple nonlinear dynamical system: The first equation is referred to as the *state equation* and the second as the *output equation*, for reasons that are clear from Figure 2.1.

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{u}(t), \quad \text{state equation,} \tag{2.3}$$

$$\mathbf{y}(t) = \mathbf{f}(\mathbf{x}(t)), \quad \text{output equation.} \tag{2.4}$$

Furthermore, from (2.2) and (2.4) the output $\mathbf{y}(t)$ is the negative of the residue $\mathbf{r}(t)$:

$$\mathbf{y}(t) = -\mathbf{r}(t). \tag{2.5}$$

The problem of finding a zero of $\mathbf{f}(\cdot)$ can now be formulated in control terms as follows. Find a control $\mathbf{u}(t)$ that will drive the output (i.e., negative of the residue) to zero and, consequently, the state variable $\mathbf{x}(t)$ to the desired solution. In terms of standard control jargon, this is a *regulation problem* where the output must regulate to (i.e., become equal to) a reference signal, which in this case is zero: a glance at Figure 2.1 makes this description clear.

If the input is regarded as arbitrary and denoted as $\mathbf{v}$, and the method (dynamical system) is now required to find $\mathbf{x}$ such that $\mathbf{f}(\mathbf{x}) = \mathbf{s}$, i.e., a trajectory $\mathbf{x}(t)$ such that $\lim_{t\to\infty} \mathbf{x}(t) = \mathbf{v}$, then the problem is referred to, in control terminology, as an asymptotic tracking problem, because the system output $\mathbf{y}$ is required to track the input $\mathbf{v}$ (see Figure 2.1). In this case, if it is also required that the method work in spite of perturbations (i.e.,

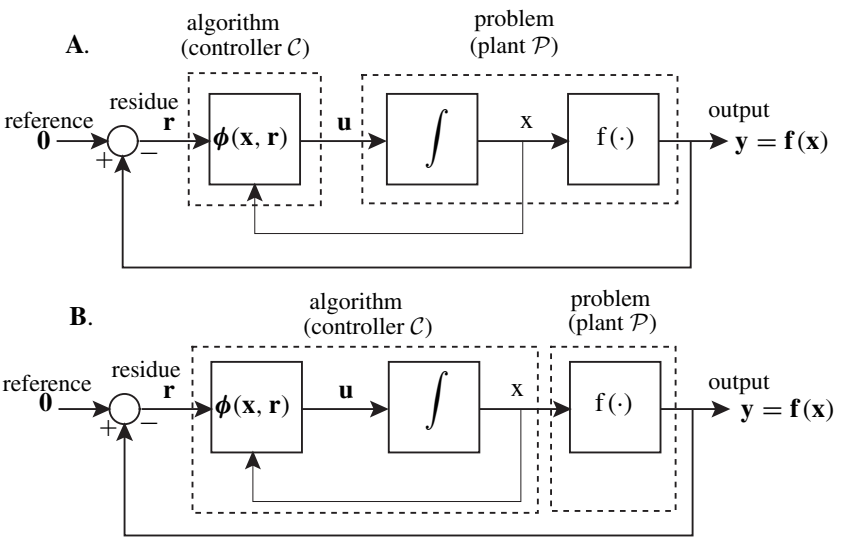

Figure 2.1. A: *A continuous realization of a general iterative method to solve the equation* $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ *represented as a feedback control system. The plant, object of the control, represents the problem to be solved, while the controller* $\boldsymbol{\phi}(\mathbf{x}, \mathbf{r})$*, a function of* $\mathbf{x}$ *and* $\mathbf{r}$*, is a representation of the algorithm designed to solve it. Thus choice of an algorithm corresponds to the choice of a controller. As quadruples,* $\mathcal{P} = \{\mathbf{0}, \mathbf{I}, \mathbf{f}, \mathbf{0}\}$ *and* $\mathcal{C} = \{\mathbf{0}, \mathbf{0}, \mathbf{0}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{r})\}$. B: *An alternative continuous realization of a general iterative method represented as a feedback control system. As quadruples,* $\mathcal{P} = \{\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{f}\}$ *and* $\mathcal{C} = \{\mathbf{0}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{r}), \mathbf{I}, \mathbf{0}\}$*. Note that* $\mathbf{x}$ *is the state vector of the plant in part A, while it is the state vector of the controller in part B.*

errors in the input or output data), then the *internal model principle* (see Chapter 1) states that this so-called robust tracking property holds if and only if the feedback system is internally stable and there is a "model" of the disturbance (to be rejected) in the feedback loop. Since the input to be tracked can be viewed as a step function, i.e., one that goes from $\mathbf{0}$ to the constant value $\mathbf{v}$, this model is required to be an integrator (see Chapter 1 for a simple derivation in the linear case). This is depicted in Figures 2.1A and 2.1B, which show an integrator considered as part of the plant and part of the controller, respectively; in both cases, however, the integrator occurs in the feedback loop and the internal model principle is satisfied.

Having introduced the idea of time-dependence of the variables $\mathbf{x}(t)$, $\mathbf{u}(t)$, etc., in what follows, in order to lighten notation, the time variable will be dropped whenever possible and the variables written simply as $\mathbf{x}$, $\mathbf{u}$, etc.

General feedback control perspective on zero finding dynamical systems

From the point of view of the regulation problem in control, a natural idea is to feed back the output variable $\mathbf{y}$ in order to drive it to the reference value of zero. This is also referred

to as *closing the loop*. It is also reasonable to expect a negative *feedback gain* that will be a function of the present "guess" of the state variable $\mathbf{x}$, as well as of the present value $\mathbf{f}(\mathbf{x}) = \mathbf{y}$. A simple approach is to choose a *feedback law* of the following type:

$$\mathbf{u} = \boldsymbol{\phi}(\mathbf{x}, \mathbf{r}), \tag{2.6}$$

leading to a so-called closed-loop system of the form

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \boldsymbol{\phi}(\mathbf{x}, \mathbf{r}). \tag{2.7}$$

Thus, the problem to be solved now is that of choosing the feedback law $\boldsymbol{\phi}(\mathbf{x}, \mathbf{r})$ in such a way as to make the closed-loop system asymptotically stable, driving the residue $\mathbf{r}$ to zero as fast as possible, thus solving the original problem of finding a solution to (2.1). In terms of Figure 2.1A, the choice (2.6) corresponds to choosing a controller

$$\mathcal{C}_s = \{\mathbf{0}, \mathbf{0}, \mathbf{0}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{r})\} \tag{2.8}$$

for a plant $\mathcal{P} = \{\mathbf{0}, \mathbf{I}, \mathbf{f}, \mathbf{0}\}$. In control terminology, controller $\mathcal{C}_s$ is referred to as a *static state-dependent controller*, since the "gain" $\boldsymbol{\phi}(\mathbf{x}, \mathbf{r})$ between controller input and output depends on the controller input $\mathbf{r}$ as well as on the plant state $\mathbf{x}$.

More generally, in the configuration of Figure 2.1A, the problem is to choose a controller $\mathcal{C}_d$ (not necessarily of the form $\{\mathbf{0}, \mathbf{0}, \mathbf{0}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{r})\}$) that regulates the plant output to zero, thus finding a zero of the function $\mathbf{f}(\cdot)$. In particular, the choice

$$\mathcal{C}_d = \{\boldsymbol{\Gamma}, \boldsymbol{\varphi}(\mathbf{x}, \mathbf{r}), \mathbf{I}, \mathbf{0}\}, \tag{2.9}$$

which is a *dynamic controller*, will be considered in what follows (see Figure 2.3), so that the combined plant-controller system is described by the equations

$$\dot{\mathbf{x}} = \mathbf{u}, \tag{2.10}$$

$$\dot{\mathbf{u}} = -\boldsymbol{\Gamma}\mathbf{u} + \boldsymbol{\varphi}(\mathbf{x}, \mathbf{r}). \tag{2.11}$$

The reader will observe that $\mathbf{u}$, in (2.11), is the controller state vector, which is also chosen as the controller output and, in turn, equal to the plant input in (2.10). The control problem for the plant-controller pair (2.10)–(2.11) is to choose the matrix $\boldsymbol{\Gamma}$ and the function $\boldsymbol{\varphi}(\mathbf{x}, \mathbf{r})$ in order to regulate the plant output to zero, thus finding a zero of the function $\mathbf{f}(\cdot)$.

In summary, it is desired to solve the problem of regulating the output to zero for a plant $\mathcal{P} = \{\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{f}\}$ and the following choices of controller: (i) $\mathcal{C}_s = \{\mathbf{0}, \mathbf{0}, \mathbf{0}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{r})\}$ and (ii) $\mathcal{C}_d = \{\boldsymbol{\Gamma}, \boldsymbol{\varphi}(\mathbf{x}, \mathbf{r}), \mathbf{I}, \mathbf{0}\}$.

The remainder of this section shows how the CLF approach can be utilized to design both controllers $\mathcal{C}_s$ and $\mathcal{C}_d$. In fact, the CLF approach does more: the structure of the controllers $\mathcal{C}_s$ and $\mathcal{C}_d$, as well as the particular choices of $\boldsymbol{\phi}(\mathbf{x}, \mathbf{r})$, $\boldsymbol{\Gamma}$, and $\boldsymbol{\varphi}(\mathbf{x}, \mathbf{r})$, emerge naturally from the CLF approach.

The reader will observe that similar controller design problems can be formulated for the plant-controller partition presented in Figure 2.1B. This book, for the most part, will consider the controller design problem for the configuration of Figure 2.1A.

CLF/LOC approach to design continuous zero finding algorithms

The objective of this section is to show how control ideas can be used in the systematic design of algorithms, and thus the problems just posed above will be solved using the CLF and *Liapunov optimizing control* (LOC) approaches. Indeed, the specific form of the feedback law (2.7) will not be assumed, but rather derived from the CLF/LOC approach which results in the choice of the control $\mathbf{u}$.

The general scheme is as follows. A candidate Liapunov function V is chosen. The time derivative $\dot{V}$ along the trajectories of the dynamical system is calculated and is a function of the state vector $\mathbf{x}$, the output $\mathbf{y} = -\mathbf{r}$, and the input $\mathbf{u}$. The CLF approach can be described as the choice of *any* $\mathbf{u}$ that makes $\dot{V}$ negative definite, while the LOC approach goes a step further in using the degrees of freedom available and demands that the input $\mathbf{u}$ be chosen in such a way as to make $\dot{V}$ as negative as possible. The attempt to do the latter involves minimizing $\dot{V}$ with respect to the (free) control variable: This leads in one direction to a connection with the so-called speed-gradient method when there are no restrictions on the control. In another direction, when a bounded control is to be applied, it is often true that the optimizing control involves a signum function of the control, and this leads to a connection with variable structure control. There are, in fact, close connections between Liapunov optimizing control, speed-gradient control, variable structure control, and optimal control, and these will be examined in more detail in this chapter as well as in Chapter 3. Another way of thinking about LOC is to regard it as a greedy approach, in the sense that it makes the best local choice of control. The controllers $\mathcal{C}_s$ and $\mathcal{C}_d$ are designed using the CLF/LOC approach.

General stability result for quadratic CLF in residue coordinates

In order to use Liapunov theory, the control design is done in terms of the residual vector $\mathbf{r}$, so that the stability analysis is carried out with respect to the equilibrium $\mathbf{r} = \mathbf{0}$. Thus, it is assumed that a local change of coordinates is possible from the variable $\mathbf{x}$ to the variable $\mathbf{r}$. Since $\mathbf{r} = -\mathbf{f}(\mathbf{x})$, by the inverse function theorem, if it is assumed that the Jacobian of $\mathbf{f}(\cdot)$ is invertible, then $\mathbf{f}$ itself is locally invertible; i.e., the desired change of coordinates exists. Accordingly, taking the time derivative of (2.2) leads to

$$\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} = -\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = -\mathbf{D}_{\mathbf{f}}(\mathbf{x})\dot{\mathbf{x}}, \tag{2.12}$$

where $\mathbf{D}_{\mathbf{f}}(\mathbf{x})$ denotes the Jacobian matrix of $\mathbf{f}$ at $\mathbf{x}$ and $\dot{\mathbf{x}}$ denotes the time derivative of $\mathbf{x}$. Notice, from (2.3), that (2.12) can also be written as

$$\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} = -\mathbf{D}_{\mathbf{f}}(\mathbf{x})\mathbf{u}. \tag{2.13}$$

One simple choice of CLF for design of the static controller $\mathcal{C}_s$ is based on the 2-norm of the residue vector $\mathbf{r}$:

$$V(\mathbf{r}) := \frac{1}{2}\|\mathbf{r}\|_2^2 = \frac{1}{2}\mathbf{r}^T\mathbf{r}, \tag{2.14}$$

which is evidently positive definite and only assumes the value 0 at the desired equilibrium $\mathbf{r} = \mathbf{0}$. The time derivative of V along the trajectories of (2.13), denoted $\dot{V}$, is given by

$$\dot{V}(\mathbf{r}) = \mathbf{r}^T\dot{\mathbf{r}}. \tag{2.15}$$

Substituting (2.13) in (2.15) yields

$$\dot{V}(\mathbf{r}) = -\mathbf{r}^T \mathbf{D}_{\mathbf{f}}(\mathbf{x})\mathbf{u}, \tag{2.16}$$

which is one possible starting point for a CLF/LOC approach to the design of the control variable $\mathbf{u}$. More precisely, in order for the system to be asymptotically stable, it is necessary to choose $\mathbf{u} = \mathbf{u}(\mathbf{r})$ in such a way that $\dot{V}$ becomes negative definite. Such a choice will prove asymptotic stability of the zero solution $\mathbf{r} \equiv \mathbf{0}$ of the closed-loop system

$$\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} = -\mathbf{D}_{\mathbf{f}}(\mathbf{x})\mathbf{u}(\mathbf{r}), \tag{2.17}$$

leading to a prototypical stability result, for the static controller case, of the following type.

Theorem 2.1. *Given a function* $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$*, suppose that* $\mathbf{x}^*$ *is a simple zero of* $\mathbf{f}$ *such that the Jacobian matrix* $\mathbf{D}_{\mathbf{f}}(\mathbf{x})$ *of* $\mathbf{f}$ *is invertible in some neighborhood* $\mathcal{N}(\mathbf{x}^*)$ *of* $\mathbf{x}^*$*. Then, for all initial conditions* $\mathbf{x}_0 \in \mathcal{N}(\mathbf{x}^*)$*, the corresponding trajectories of the dynamical system*

$$\dot{\mathbf{x}} = \mathbf{u}(\mathbf{r}), \quad \mathbf{x}(0) = \mathbf{x}_0 \in \mathcal{N}(\mathbf{x}^*), \tag{2.18}$$

where $\mathbf{u}(\mathbf{r})$ *is a feedback control chosen by the CLF/LOC method, in the manner specified in Table* 2.1*, converge to the zero* $\mathbf{x}^*$ *of* $\mathbf{f}(\cdot)$.

Note that Theorem 2.1 yields a local stability result, since its proof depends on the stability properties of the residue system (2.17), which is obtained after a local change of coordinates. The stability type—exponential, asymptotic, or finite time—depends on the particular control that is chosen and will be discussed further below.

CLF/LOC design of continuous algorithms with static controllers $\mathcal{C}_s$

The CLF/LOC approach, applied to (2.16), is now used in order to derive the choices of $\mathbf{u}(\mathbf{r})$, leading to a proof of Theorem 2.1. Note that if

$$\mathbf{u} = \mathbf{D}_{\mathbf{f}}^{-1}\mathbf{P}\mathbf{r},$$

then, from (2.16),

$$\dot{V} = -\mathbf{r}^T\mathbf{P}\mathbf{r},$$

which is clearly negative definite, for any choice of positive definite matrix $\mathbf{P}$. The resulting continuous algorithm is written as follows:

$$\dot{\mathbf{x}} = -\mathbf{D}_{\mathbf{f}}^{-1}\mathbf{P}\mathbf{f}(\mathbf{x}). \tag{2.19}$$

This dynamical system, for $\mathbf{P} = \mathbf{I}$, is known as the *continuous Newton (CN) algorithm* and is well known in the literature (see [Alb71, Neu99] and the references therein).

We now show how the CLF/LOC approach can be exploited to go beyond the CN algorithm. Taking another look at (2.16), it is clear that if we choose

$$\mathbf{u} = \mathbf{D}_{\mathbf{f}}^{-1}\mathrm{sgn}(\mathbf{r}), \tag{2.20}$$

where sgn is the signum function, then from (2.16),

$$\dot{V} = -\mathbf{r}^T \mathrm{sgn}(\mathbf{r}) = -\|\mathbf{r}\|_1,$$

which is clearly negative definite and leads to a new algorithm, written as follows:

$$\dot{\mathbf{x}} = -\mathbf{D}_\mathbf{f}^{-1} \mathrm{sgn}(\mathbf{f}(\mathbf{x})). \tag{2.21}$$

The above dynamical system is called a *Newton variable (NV) structure algorithm*, and its notable feature is that the right-hand side of (2.21) is discontinuous, so that the Filippov solution concept and associated stability theory discussed in section 1.5 must be used.

A third choice that suggests itself is

$$\mathbf{u} = \mathbf{P}\mathbf{D}_\mathbf{f}^T \mathbf{r}$$

leading, from (2.16), to

$$\dot{V} = -\mathbf{r}^T \mathbf{D}_\mathbf{f} \mathbf{P} \mathbf{D}_\mathbf{f}^T \mathbf{r},$$

which is clearly negative definite, under the hypotheses of Theorem 2.1. The resulting continuous algorithm is written as follows:

$$\dot{\mathbf{x}} = -\mathbf{P}\mathbf{D}_\mathbf{f}^T \mathbf{f}(\mathbf{x}). \tag{2.22}$$

This is again a new algorithm, which will be referred to as the *continuous Jacobian matrix transpose (CJT) algorithm*.

A fourth choice is

$$\mathbf{u} = \mathrm{sgn}(\mathbf{D}_\mathbf{f}^T \mathbf{r}), \tag{2.23}$$

which yields, from (2.16),

$$\dot{V} = -(\mathbf{D}_\mathbf{f}^T \mathbf{r})^T \mathrm{sgn}(\mathbf{D}_\mathbf{f}^T \mathbf{r}) = -\|\mathbf{D}_\mathbf{f}^T \mathbf{r}\|_1,$$

which again is negative definite under the nonsingularity hypothesis on $\mathbf{D}_\mathbf{f}$. Note that this choice is a Liapunov optimizing control, since it makes $\dot{V}$ as negative as possible, under the constraint that the 1-norm of the control should not exceed unity. The corresponding algorithm is written as follows:

$$\dot{\mathbf{x}} = -\mathrm{sgn}(\mathbf{D}_\mathbf{f}^T \mathbf{f}(\mathbf{x})). \tag{2.24}$$

This algorithm will be referred to as a *variable structure Jacobian transpose (VJT) algorithm*. Once again, the right-hand side of (2.24) is discontinuous, so the Filippov solution concept and associated stability theory (section 1.5) should be used.

Finally, as an illustration of how the choice of Liapunov function influences the resulting continuous algorithm, instead of using the 2-norm as a CLF, let the 1-norm

$$W(\mathbf{r}) = \|\mathbf{r}\|_1 = \mathbf{r}^T \mathrm{sgn}(\mathbf{r}) = \mathrm{sgn}(\mathbf{r})^T \mathbf{r} \tag{2.25}$$

be chosen as a nonsmooth CLF. In this case, the signum function is constant except at the origin, so that, formally speaking [Utk92], its derivative is zero, except at the origin, and we can write

$$\dot{W}(\mathbf{r}) = \mathrm{sgn}(\mathbf{r})^T \dot{\mathbf{r}} = \mathrm{sgn}(\mathbf{r})^T (-\mathbf{D}_\mathbf{f} \mathbf{u}),$$

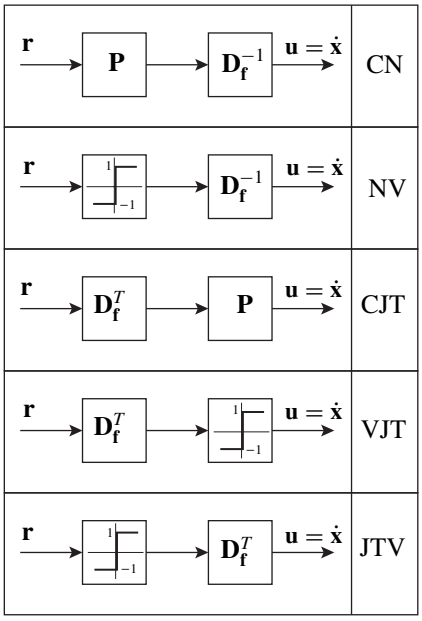

Figure 2.2. *The structure of the CLF/LOC controllers* $\boldsymbol{\phi}(\mathbf{x}, \mathbf{r})$: *The block labeled* $\mathbf{P}$ *corresponds to multiplication by a positive definite matrix* $\mathbf{P}$*, the blocks labeled* $\mathbf{D}_\mathbf{f}^T$ *and* $\mathbf{D}_\mathbf{f}^{-1}$ *depend on* $\mathbf{x}$ *(see Figure* 2.1A *and Table* 2.1).

using (2.13). Thus, the choice

$$\mathbf{u} = \mathbf{D}_\mathbf{f}^T \operatorname{sgn}(\mathbf{r}) \tag{2.26}$$

gives

$$\dot{W}(\mathbf{r}) = -\operatorname{sgn}(\mathbf{r})^T (\mathbf{D_f} \mathbf{D}_\mathbf{f}^T) \operatorname{sgn}(\mathbf{r}) = -\|\mathbf{D}_\mathbf{f}^T \operatorname{sgn}(\mathbf{r})\|_2^2,$$

which is clearly negative definite under the nonsingularity hypothesis on $\mathbf{D_f}$. The resulting algorithm is written as follows:

$$\dot{\mathbf{x}} = -\mathbf{D}_\mathbf{f}^T \operatorname{sgn}(\mathbf{f}(\mathbf{x})). \tag{2.27}$$

This algorithm is also new and will be referred to as a *Jacobian matrix transpose variable structure (JTV) algorithm* and, once again, the right-hand side is discontinuous, so the Filippov theory of section 1.5 should be used. The controller block, $\boldsymbol{\phi}(\mathbf{x}, \mathbf{r})$ (Figure 2.1), corresponding to each of the five algorithms, is shown in Figure 2.2.

At this point the observant reader might well ask where the LOC idea is being used, specifically in the design of the CN, NV, CJT, and JTV algorithms, since so far, LOC has been used only in the VJT method.

In order to answer this question, observe that there is flexibility in the choice of the plant. For example, suppose that, instead of (2.3), we choose the plant

$$\dot{\mathbf{x}} = \mathbf{D}_\mathbf{f}^{-1} \mathbf{u}, \tag{2.28}$$

which is easily seen to be the system

$$\dot{\mathbf{r}} = -\mathbf{u}, \tag{2.29}$$

Table 2.1. *Choices of* $\mathbf{u}(\mathbf{r})$ *in Theorem* 2.1 *that result in stable zero finding dynamical systems.*

$\mathbf{u}(\mathbf{r})$	$\dot{V}$ (rows 1–4) $\dot{W}$ (row 5)	Continuous algorithm	Acronym	Equation number
$\mathbf{D_f^{-1}Pr}$,	$-\mathbf{r}^T\mathbf{Pr}$	$\dot{\mathbf{x}} = -\mathbf{D_f^{-1}Pf(x)}$	CN	(2.19)
$\mathbf{D_f^{-1}}\mathrm{sgn}(\mathbf{r})$,	$-\|\mathbf{r}\|_1$	$\dot{\mathbf{x}} = -\mathbf{D_f^{-1}}\mathrm{sgn}(\mathbf{f(x)})$	NV	(2.21)
$\mathbf{PD_f^T r}$	$-\mathbf{r}^T\mathbf{D_f PD_f^T r}$	$\dot{\mathbf{x}} = -\mathbf{PD_f^T f(x)}$	CJT	(2.22)
$\mathrm{sgn}(\mathbf{D_f^T r})$	$-\|\mathbf{D_f^T r}\|_1$	$\dot{\mathbf{x}} = -\mathrm{sgn}(\mathbf{D_f^T f(x)})$	VJT	(2.24)
$\mathbf{D_f^T}\,\mathrm{sgn}(\mathbf{r})$	$-\|\mathbf{D_f^T}\,\mathrm{sgn}(\mathbf{r})\|_2^2$	$\dot{\mathbf{x}} = -\mathbf{D_f^T}\,\mathrm{sgn}(\mathbf{f(x)})$	JTV	(2.27)

in $\mathbf{r}$-coordinates. Maintaining the output equation (2.4) and the 2-norm Liapunov function V in (2.14), the time derivative of V along the trajectories of (2.28) is given by

$$\dot{V} = -\mathbf{r}^T\mathbf{D_f}\dot{\mathbf{x}} = -\mathbf{r}^T\mathbf{D_f D_f^{-1}u} = -\mathbf{r}^T\mathbf{u}. \tag{2.30}$$

Now, for (2.30), the choice $\mathbf{u} = \mathrm{sgn}(\mathbf{r})$ in (2.20) is clearly an LOC, under the constraint that all components of the control vector should be less than or equal to unity in absolute value (i.e., $\|\mathbf{u}\|_\infty = 1$). This yields the Newton variable structure method (2.21). Furthermore, the choice $\mathbf{u} = \mathbf{r}$ in (2.30), which leads to the CN method (2.19), is also optimal in the sense of being the simplest choice (of state feedback) that leads to an exponentially stable system in $\mathbf{r}$-coordinates,

$$\dot{\mathbf{r}} = -\mathbf{r}. \tag{2.31}$$

Similarly, consider the alternative choice of plant,

$$\dot{\mathbf{x}} = \mathbf{D_f^T u}. \tag{2.32}$$

Then, maintaining the output equation (2.4) and using the 1-norm Liapunov function W in (2.25), the time derivative of W along the trajectories of (2.32) is given by

$$\dot{W} = \mathrm{sgn}(\mathbf{r})^T\dot{\mathbf{r}} = -\mathrm{sgn}(\mathbf{r})^T\mathbf{D_f}\dot{\mathbf{x}} = -\mathrm{sgn}(\mathbf{r})^T\mathbf{D_f D_f^T u}. \tag{2.33}$$

Now the choice $\mathbf{u} = \mathrm{sgn}(\mathbf{r})$ is clearly a possible LOC choice that gives the JTV algorithm in (2.27). If, instead of W, V is used, then CJT is seen to result from the simple choice $\mathbf{u} = \mathbf{r}$.

In summary, the CLF/LOC approach has been shown to lead, at least, to the five different algorithms considered above. For ease of reference, these zero finding dynamical systems are organized in Table 2.1. Each system is given an acronym formed using the following conventions: N = Newton, C = continuous-time, V = variable structure, JT = Jacobian matrix transpose.

The first row, for $\mathbf{P} = \mathbf{I}$, corresponds to the well-known and much studied CN method, so called because it is clear that a discretization of $\dot{\mathbf{x}} = -\mathbf{D_f^{-1}f(x)}$, by the forward Euler method, will lead to the classical Newton–Raphson method, discussed further in section

2.2. It has several notable properties [HN05], one of which is singled out here for mention: Convergence to the zero is exponential. This is easily seen in the closed-loop residue system, where choosing $\mathbf{P} = \alpha\mathbf{I}$, $\alpha > 0$ gives

$$\dot{\mathbf{r}} = -\alpha\mathbf{r}, \quad \mathbf{r}_0 = -\mathbf{f}(\mathbf{x}_0), \tag{2.34}$$

which has the solution $\mathbf{r}(t) = e^{-\alpha t}\mathbf{r}_0$. The choice of α clearly affects the speed of convergence of the residue to zero and can be regarded as an additional design parameter. Having said this, from now on, unless otherwise stated, we will take $\mathbf{P}$ to be $\mathbf{I}$, to simplify matters. Another way of looking at the exponential stability of the zero solution of (2.19) is to observe that it can be written in the so-called integrated form by avoiding the inversion of the Jacobian matrix [Neu99]. In other words, (2.19) is first written as $\mathbf{D_f}\dot{\mathbf{x}} = -\mathbf{f}(\mathbf{x})$ and then, since $\dot{\mathbf{f}} = \mathbf{D_f}\dot{\mathbf{x}}$, finally

$$\dot{\mathbf{f}} = -\mathbf{f}, \quad \mathbf{f}_0 = \mathbf{f}(\mathbf{x}_0). \tag{2.35}$$

It should also be noted that, as far as the dynamical systems for zero finding are concerned, the variables $\mathbf{r}$ and $\mathbf{f}$ are equivalent: note the equivalence of (2.34) with $\alpha = 1$ and (2.35). Said differently, it is equivalent to work in $\mathbf{r}$-coordinates or $\mathbf{f}$-coordinates, up to a change of sign.

The second, fourth, and fifth rows correspond to *variable structure methods*, with some new features, the most notable one being that they lead to differential equation with discontinuous right-hand sides. It is necessary to be careful about existence, uniqueness, and stability issues for differential equations with discontinuous right-hand sides, but for reasons of organization and brevity (of this chapter), formal manipulations of Liapunov functions, smooth and nonsmooth, have been used to motivate the choices made. The reader should be aware that these formal manipulations can be made rigorous, and the basic ideas of one approach, due to Filippov, are outlined in Chapter 1, where further references are also supplied.

As pointed out at the beginning of this section, a general observation that should be made about the CLF and LOC approaches is that they lead to the form of feedback law that one expects intuitively, without the need to prespecify the form of the feedback law as (2.6), thus showing the power of the Liapunov approach. An additional demonstration of this power lies in the fact that it serves to unify several different types of algorithms, as well as providing a method of generating new algorithms. One way of doing the latter is to use different Liapunov functions, and an example of a choice different from (2.14) occurs in the fifth row of Table 2.1, as well as in the design of dynamic controllers, discussed further ahead in this section.

Gradient control perspective on zero finding algorithms

The static controller-based algorithms in Table 2.1 can also be regarded as examples of gradient control. To do this, we recall (2.17):

$$\dot{\mathbf{r}} = -\mathbf{D_f}\mathbf{u}. \tag{2.36}$$

For the Liapunov function (2.14),

$$\nabla V(\mathbf{r}) = \mathbf{r}. \tag{2.37}$$

Now, substituting the choices of $\mathbf{u}$ from rows 1 and 3 of Table 2.1 in (2.36), and using (2.37), yields, respectively,

$$\dot{\mathbf{r}} = -\mathbf{P}\nabla V(\mathbf{r}) \tag{2.38}$$

and

$$\dot{\mathbf{r}} = -\mathbf{D_f}\mathbf{P}\mathbf{D_f}^T \nabla V(\mathbf{r}), \tag{2.39}$$

which are both gradient systems in the terminology of Chapter 1 (for which V is a Liapunov function) since the matrices $\mathbf{P}$ and $\mathbf{D_f}\mathbf{P}\mathbf{D_f}^T$ are positive definite, by choice and by hypothesis, respectively. Thus the controls chosen in rows 1 and 3 can be viewed as *gradient controls* of the system (2.36).

Substitution of the control choices in rows 2 and 5 of Table 2.1 in (2.36) leads, respectively, to the following systems:

$$\dot{\mathbf{r}} = -\text{sgn}(\mathbf{r}) \tag{2.40}$$

and

$$\dot{\mathbf{r}} = -\mathbf{D_f}\mathbf{D_f}^T \text{sgn}(\mathbf{r}). \tag{2.41}$$

These systems are both in Persidskii-type form with discontinuous right-hand side, for which asymptotic stability of the zero solution $\mathbf{r} \equiv \mathbf{0}$ is ensured. Furthermore, as pointed out in Chapter 1, these particular Persidskii systems can also be written as GDSs with discontinuous right-hand sides, so that, once again, these entries in Table 2.1 can be viewed as examples of gradient control.

The observation that (2.40) is in Persidskii-type form permits a slight generalization of the second row, by making the choice $\mathbf{u} = \mathbf{D_f}^{-1}\mathbf{A}\text{sgn}(\mathbf{r})$, where $\mathbf{A}$ is a diagonally stable matrix, since this will lead to the asymptotically stable Persidskii-type system (see Chapter 1)

$$\dot{\mathbf{r}} = -\mathbf{A}\text{sgn}(\mathbf{r}), \tag{2.42}$$

with the corresponding generalized variable structure Newton method being

$$\dot{\mathbf{x}} = -\mathbf{D_f}^{-1}\mathbf{A}\,\text{sgn}(\mathbf{f}). \tag{2.43}$$

Continuous algorithms derived from a speed gradient perspective

The speed gradient method [FP98], reviewed in Chapter 1, can be used to systematize Liapunov optimizing control design in the context of the zero finding problem, as well as lead to the design of new algorithms.

In order to do this, consider the special affine nonlinear system given by (2.3) and (2.4) and let the CLF be chosen as in (2.14). Then, clearly

$$\dot{V}(\mathbf{r}) = \mathbf{r}^T\dot{\mathbf{r}} = -\mathbf{r}^T\mathbf{D_f}\mathbf{u}$$

and

$$\nabla_{\mathbf{u}}\dot{V}(\mathbf{r}) = -\mathbf{D_f}^T\mathbf{r}.$$

Using (1.126) and (1.128), with $\Gamma = \gamma\mathbf{I}$, yields CJT in Table 2.1, with $\mathbf{P} = \gamma\mathbf{I}$. Similarly, using (1.126) and (1.129), we arrive at VJT.

Starting with the dynamical system (2.28), and using the 2-norm CLF $V(\mathbf{r})$, it is easy to check that $\nabla_{\mathbf{u}}\dot{V} = \mathbf{r}$, so that using (1.126) and (1.128) yields CN, while using (1.126) and (1.129) leads to NV. Finally, using the 1-norm W, it is possible to recover JTV. A disadvantage of using the speed gradient method is that it requires some stringent hypotheses (see [FP98]). However, given the close connections between the CLF/LOC and the speed gradient approach, it is of interest to point out this possible route to the algorithms in Table 2.1.

Benchmark examples for zero finding algorithms

Two examples that are used as benchmarks to test the qualitative behavior of the various numerical algorithms proposed in this chapter, as well as elsewhere in the book, are given below.

The Rosenbrock function [Ros60] is a famous example of a difficult optimization problem and is defined as follows:

$$\mathbf{f}(\mathbf{x}) : \mathbb{R}^2 \to \mathbb{R}^2 : (x_1, x_2) \mapsto a(x_1^2 - x_2)^2 + (x_1 - b)^2, \tag{2.44}$$

where the parameters a and b are to be specified. The problem of finding a minimum of this function can be expressed as the zero finding problem for the gradient of $\mathbf{f}$, i.e., by finding the zeros of $\mathbf{g} = \nabla\mathbf{f}$, where

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \partial\mathbf{f}/\partial x_1 \\ \partial\mathbf{f}/\partial x_2 \end{bmatrix} = \begin{bmatrix} 4ax_1(x_1^2 - x_2) + 2(x_1 - b) \\ -2a(x_1^2 - x_2) \end{bmatrix}. \tag{2.45}$$

The second example, due to Branin [Bra72], inspired by a tunnel diode circuit and used as a benchmark example for many zero finding algorithms, is the problem of finding the zeros of functions $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$, where

$$\begin{aligned} f_1(x_1, x_2) &= a - bx_2 + c\sin(dx_2) - x_1, \\ f_2(x_1, x_2) &= x_2 - e\sin(fx_1). \end{aligned} \tag{2.46}$$

The values of the parameters, used in [Bra72] and several other papers that cite it, are $a = 1$, $b = 2$, $d = 4\pi$, $e = 0.5$, $f = 2\pi$; the value of c determines the number of zeros of the system $f_1(x_1, x_2) = f_2(x_1, x_2) = 0$. For example, for $c = 0$, the system has 5 zeros; for $c = 0.5$, the system has 7 zeros; for $c = 1$, the system has 15 zeros.

Changing the singularities of the Newton vector field

An important feature of the continuous Newton method, defined by the Newton vector field associated to a function $\mathbf{f}(\cdot)$,

$$\mathbf{N}_{\mathbf{f}} = -\mathbf{D}_{\mathbf{f}}^{-1}\mathbf{f}(\mathbf{x}), \tag{2.47}$$

is that the singularities of $\mathbf{N}_{\mathbf{f}}$ may occur in locations different from the desired zeros of $\mathbf{f}$. More specifically, convergence of trajectories of the CN method to the desired zeros (of $\mathbf{f}$) could be adversely affected by the presence of the so-called extraneous singularities [Bra72, Gom75, RZ99, ZG02], defined below.

Let the *set of zeros* of the function $\mathbf{f}$ be denoted as

$$\mathcal{Z} := \{\mathbf{x} : \mathbf{f}(\mathbf{x}) = \mathbf{0}\}. \tag{2.48}$$

The set of *singular points* is defined as the set of points

$$\mathcal{S} := \{\mathbf{x} : \det \mathbf{D_f}(\mathbf{x}) = 0\}. \tag{2.49}$$

Singular zeros are points in $\mathcal{S} \cap \mathcal{Z}$. The remaining zeros are called *regular*.

In particular, Zufiria, Guttalu, and coworkers [RZ99, ZG02] define a taxonomy of singularities of the Newton vector field and analyze most of the types that could occur. Rather than detail these results here, we point out that the analysis is based on the fact that the Newton vector field can be written as

$$\mathbf{N_f} = -\frac{(\operatorname{adj} \mathbf{D_f})\mathbf{f}(\mathbf{x})}{\det \mathbf{D_f}} =: \frac{1}{h(\mathbf{x})}\mathbf{g}(\mathbf{x}), \tag{2.50}$$

where $\operatorname{adj} \mathbf{D_f}$ is the classical adjoint of the Jacobian matrix $\mathbf{D_f}$. *Essential singularities* occur when the denominator $h(\mathbf{x})$ becomes zero, but the numerator $\mathbf{g}(\mathbf{x})$ does not become a zero vector. *Nonessential singularities* occur when $h(\mathbf{x}) = 0$ and $\mathbf{g}(\mathbf{x}) = \mathbf{0}$. Within the class of nonessential singularities, there is the class of *extraneous singularities* defined as

$$\mathcal{E} := \{\mathbf{x} \in \mathcal{S} : \mathbf{f}(\mathbf{x}) \neq \mathbf{0}, (\operatorname{adj} \mathbf{D_f})\mathbf{f}(\mathbf{x}) = \mathbf{0}\}. \tag{2.51}$$

Although the Newton vector field is constructed for locating the regular and singular zeros of $\mathbf{f}$, extraneous singularities have been shown to possess great importance in different methods designed for root finding [Bra72, Gom75, RZ99, ZG02].

From this discussion, it becomes clear that for the new methods introduced and the corresponding vector fields (given by the right-hand sides of the systems in column 3, rows 2 through 5 of Table 2.1), the structure of the singularities of the vector fields changes with respect to the Newton vector field. In particular, it becomes clear that although much research has focused on "desingularizing" and continuous extensions of the Newton vector field in the presence of the different types of singularities [DS75, DK80a, DK80b, Die93, RZ99, ZG02], control theory provides an alternative approach in which the flexibility in the choice of feedback control allows the algorithm designer to change the singularity structure of the vector field.

Example 2.2. Branin [Bra72] used Rosenbrock's example (2.44) to illustrate the problem of extraneous singularities of the Newton vector field. We use this example here to show that, while the Newton vector field has an extraneous singularity that affects trajectories of the CN dynamical system, on the other hand, the NV vector field does not possess this extraneous singularity. As a consequence, NV trajectories, for some initial conditions, are better behaved than CN trajectories, exhibiting faster and more consistent progress to the solution.

It is easy to check that $(x_1, x_2) = (b, b^2)$ is the solution of $\mathbf{g} = \mathbf{0}$ in (2.45). Furthermore, the Jacobian matrix $\mathbf{D_g}$ is easily calculated as follows:

$$\mathbf{D_g}(\mathbf{x}) = \begin{bmatrix} 4a(3x_1^2 - x_2) + 2 & -4ax_1 \\ -4ax_1 & 2a \end{bmatrix}. \tag{2.52}$$

Its classical adjoint $\operatorname{adj} \mathbf{D_g}$ is

$$\operatorname{adj} \mathbf{D_g}(\mathbf{x}) = \begin{bmatrix} 2a & 4ax_1 \\ 4ax_1 & 4a(3x_1^2 - x_2) + 2 \end{bmatrix}. \tag{2.53}$$

It is now easy to verify that

$$\mathbf{x}_{\text{ex}}^{\text{ctn}} = \begin{bmatrix} b \\ b^2 + \frac{1}{2a} \end{bmatrix}$$

satisfies all the conditions of an extraneous singularity, namely:

$$\operatorname{adj} \mathbf{D_g}(\mathbf{x}_{\text{ex}}^{\text{ctn}}) \text{ is singular,}$$

$$\mathbf{g}(\mathbf{x}_{\text{ex}}^{\text{ctn}}) = \begin{bmatrix} -2b \\ 1 \end{bmatrix} \neq 0,$$

and

$$\operatorname{adj} \mathbf{D_g}(\mathbf{x}_{\text{ex}}^{\text{ctn}}) \mathbf{g}(\mathbf{x}_{\text{ex}}^{\text{ctn}}) = \mathbf{0}.$$

Now, if the extraneous singularity analysis is carried out for the NV vector field $\mathbf{D_g}^{-1} \operatorname{sgn}(\mathbf{g})$, all the above equations hold, except for the last one, i.e.,

$$\operatorname{adj} \mathbf{D_g}(\mathbf{x}_{\text{ex}}^{\text{ctn}}) \operatorname{sgn}(\mathbf{g}(\mathbf{x}_{\text{ex}}^{\text{ctn}})) = \operatorname{adj} \mathbf{D_g}(\mathbf{x}_{\text{ex}}^{\text{ctn}}) \operatorname{sgn} \begin{bmatrix} -2b \\ 1 \end{bmatrix} = \operatorname{adj} \mathbf{D_g}(\mathbf{x}_{\text{ex}}^{\text{ctn}}) \begin{bmatrix} -1 \\ 1 \end{bmatrix} \neq \mathbf{0},$$

showing that $\mathbf{x}_{\text{ex}}^{\text{ctn}}$ is not an extraneous singularity for the NV vector field. In other words, it is indeed possible to change the singularity structure of the Newton vector field by making the alternative choices in rows 2 through 5 of Table 2.1. Trajectories of the CN and NV methods are compared in Figure 2.4 for the parameter values $a = 0.5, b = 1$.

Branin [Bra72] showed that, as c increases from zero, the Newton vector field (2.47) for the function defined in (2.46) correspondingly possesses an increasing number of extraneous singularities.

CLF design of continuous algorithms with dynamic controllers $\mathcal{C}_d$

Starting once again from (2.3), (2.4), and (2.13), a new quadratic CLF is chosen in order to design dynamic controllers of the type $\mathcal{C}_d$ presented in (2.11). Defining

$$V_2 := \frac{1}{2}\mathbf{r}^T\mathbf{r} + \frac{1}{2}\mathbf{u}^T\mathbf{u}, \tag{2.54}$$

it is clear that $V_2(\cdot)$ is a positive definite function of the variables $\mathbf{u}$ and $\mathbf{r}$. Taking the time derivative of V_2 yields

$$\dot{V}_2 = \mathbf{r}^T\dot{\mathbf{r}} + \mathbf{u}^T\dot{\mathbf{u}} = \mathbf{u}^T(\mathbf{D_f}^T\mathbf{f} + \dot{\mathbf{u}}). \tag{2.55}$$

This simple calculation motivates some choices of "stabilizing" dynamics for $\mathbf{u}$ which are analyzed below. The most straightforward choice to make $\dot{V}_2 \leq 0$ is

$$\dot{\mathbf{u}} = -\mathbf{\Gamma}\mathbf{u} - \mathbf{D_f}^T\mathbf{f}, \qquad \mathbf{\Gamma} = \mathbf{\Gamma}^T > 0, \tag{2.56}$$

which can be justified as follows. Substituting the choice (2.56) in (2.55) gives $\dot{V}_2 = -\mathbf{u}^T\mathbf{\Gamma}\mathbf{u} \leq 0$, which is negative semidefinite, since both $\mathbf{u}$ and $\mathbf{r}$ are now state variables and

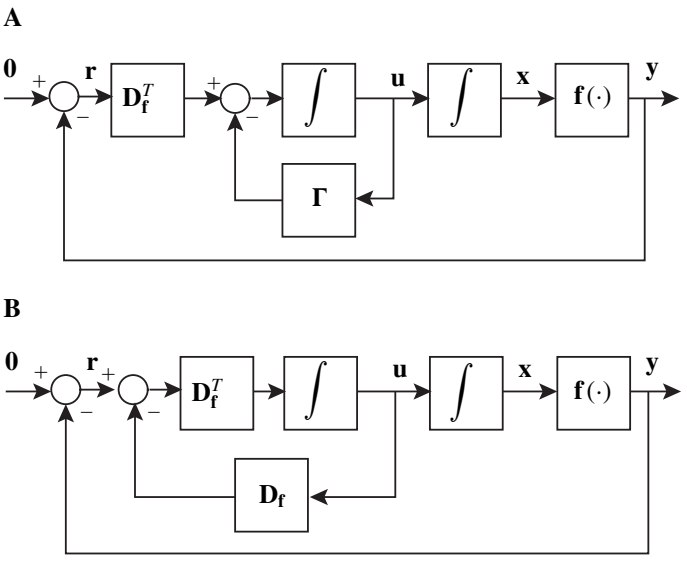

Figure 2.3. *A: Block diagram representations of continuous algorithms for the zero finding problem, using the dynamic controller defined by* (2.56)*. B: With the particular choice* $\mathbf{\Gamma} = \mathbf{D}_\mathbf{f}^T \mathbf{D_f}$.

Table 2.2. *Zero finding continuous algorithms with dynamic controllers designed using the quadratic CLF* (2.54) *(p.d. = positive definite).*

Plant	Dynamic controller	Choice of $\mathbf{\Gamma}$	$\boldsymbol{\varphi}(\mathbf{x},\mathbf{r})$	Acronym
$\dot{\mathbf{x}} = \mathbf{u}$	$\dot{\mathbf{u}} = -\mathbf{\Gamma u} - \mathbf{D}_\mathbf{f}^T \mathbf{f}(\mathbf{x})$	Arbitrary p.d.	$\mathbf{D}_\mathbf{f}^T \mathbf{r}$	DC1
$\dot{\mathbf{x}} = \mathbf{u}$	$\dot{\mathbf{u}} = -\mathbf{D}_\mathbf{f}^T (\mathbf{D_f u} + \mathbf{f}(\mathbf{x}))$	$\mathbf{D}_\mathbf{f}^T \mathbf{D_f}$	$\mathbf{D}_\mathbf{f}^T \mathbf{r}$	DC2
$\dot{\mathbf{x}} = \mathbf{D}_\mathbf{f}^{-1} \mathbf{u}$	$\dot{\mathbf{u}} = -\mathbf{\Gamma u} - \mathbf{f}(\mathbf{x})$	Arbitrary p.d.	$\mathbf{r}$	DC3
$\dot{\mathbf{x}} = \mathbf{D}_\mathbf{f}^T \mathbf{u}$	$\dot{\mathbf{u}} = -\mathbf{\Gamma u} - \mathbf{D_f D}_\mathbf{f}^T \mathbf{f}(\mathbf{x})$	Arbitrary p.d.	$\mathbf{D}_\mathbf{f}^T \mathbf{D_f r}$	DC4

$\dot{V}_2$ does not depend on the state variable $\mathbf{r}$. In this situation, LaSalle's theorem (Theorem 1.27) can be applied. In fact, $\dot{V}_2 = 0$ implies that $\mathbf{u} = \mathbf{0}$ and thus $\dot{\mathbf{u}} = \mathbf{0}$. From (2.56), it follows that $\mathbf{D}_\mathbf{f}^T \mathbf{f} = \mathbf{0}$. Now, under the usual assumption that the matrix $\mathbf{D_f}$ is invertible, it follows that $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, as desired.

One choice of $\mathbf{\Gamma}$ is $\mathbf{D}_\mathbf{f}^T \mathbf{D_f}$, which also allows a simple implementation of the resulting dynamic controller (see Figure 2.3B).

The block diagrams for these continuous algorithms are given in Figure 2.3.

Similarly, using (2.28) and (2.32), the quadratic CLF (2.54), and LaSalle's theorem, two other continuous algorithms for zero finding are easily derived. For convenience, all these algorithms are displayed in Table 2.2. Once again, it should be observed that both the controller structure and the choices that define the specific controller emerge naturally from the CLF approach and do not need to be specified a priori.

The prototypical stability result for the class of dynamical controllers (2.11) can be stated as follows.

Theorem 2.3. *Given a function* $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$*, suppose that* $\mathbf{x}^*$ *is a zero of* $\mathbf{f}$ *such that the Jacobian matrix* $\mathbf{D_f}(\mathbf{x})$ *of* $\mathbf{f}$ *is invertible in some neighborhood* $\mathcal{N}(\mathbf{x}^*)$ *of* $\mathbf{x}^*$*. Then, for all initial conditions* $\mathbf{x}_0 \in \mathcal{N}(\mathbf{x}^*)$*, the corresponding trajectories (in* $\mathbf{x}(\cdot)$*) of the dynamical system* (2.10)–(2.11) *(repeated here for ease of reference)*

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{u},\\ \dot{\mathbf{u}} &= -\boldsymbol{\Gamma}\mathbf{u} + \boldsymbol{\varphi}(\mathbf{x}, \mathbf{r}),\end{aligned}$$

where $\boldsymbol{\Gamma}$ *and* $\boldsymbol{\varphi}(\mathbf{x}, \mathbf{r})$ *are chosen by the CLF approach in the manner specified in Table* 2.2*, converge to the zero* $\mathbf{x}^*$ *of* $\mathbf{f}(\cdot)$.

A final commentary on the class of dynamic controllers is that they lead to second-order dynamical systems whose trajectories converge to the zeros of a function. To see this, one can eliminate $\mathbf{u}$ from the pair of equations (2.10)–(2.11), which yields

$$\ddot{\mathbf{x}} + \boldsymbol{\Gamma}\dot{\mathbf{x}} + \boldsymbol{\varphi}(\mathbf{x}, \mathbf{r}) = \mathbf{0}, \tag{2.57}$$

where $\boldsymbol{\varphi}(\mathbf{x}, \mathbf{r})$ is chosen in the manner specified in Table 2.2.

This idea of using second-order dynamical systems has been proposed before, having been studied in [Pol64] in an optimization context, and subsequently in [Bog71, IPZ79, DS80] in a zero finding context. All these authors, however, arrived at second-order dynamical systems using classical mechanics analogies, such as that of a heavy ball moving in a force field and subject to friction (or some other dissipative force). The discussion above shows that second-order dynamical systems for zero finding can be derived in a natural way from the CLF approach. Further remarks are made below, in an optimization context, in section 2.3.3.

Numerical simulations of continuous algorithms

Some numerical simulations of the continuous algorithms derived above are presented in Figures 2.4 and 2.5 for the Rosenbrock function; in keeping with the "perspectives" nature of this book, these are illustrative examples, and further research needs to be done in order to determine the effectiveness of the new algorithms, beyond the fact that the singularity structure of the associated vector fields is different (from that of the Newton algorithm) and, as a consequence, the trajectories take different routes to the zeros to which they converge.

2.2 Iterative Zero Finding Algorithms as Discrete-Time Dynamical Systems

> *Increasingly often, it is not optimal to try to solve a problem exactly in one pass; instead, solve it approximately, then iterate ... iterative, infinite algorithms are sometimes better... our central mission is to compute quantities that are typically uncomputable, from an analytical point of view, and to do it with lightning speed.*
>
> –L. N. Trefethen [Tre92]

We now turn to discrete or iterative algorithms, maintaining our feedback control point of view. From (2.4) and (2.5), given $\mathbf{x}_k$ at the kth iteration, we can define

$$\mathbf{r}_k := \mathbf{r}(\mathbf{x}_k) = -\mathbf{f}(\mathbf{x}_k). \tag{2.58}$$

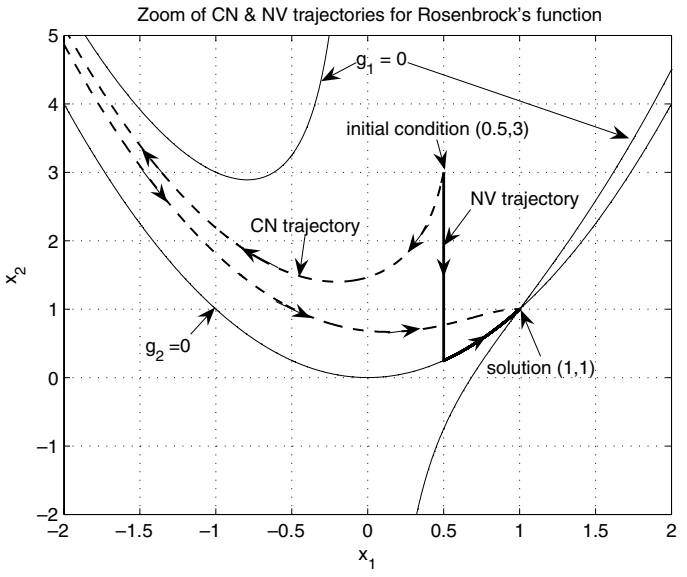

Figure 2.4. *Comparison of CN and NV trajectories for minimization of Rosenbrock's function* (2.44)*, with* $a = 0.5$*,* $b = 1$*, or equivalently, finding the zeros of* $\mathbf{g}$ *in* (2.45)*.*

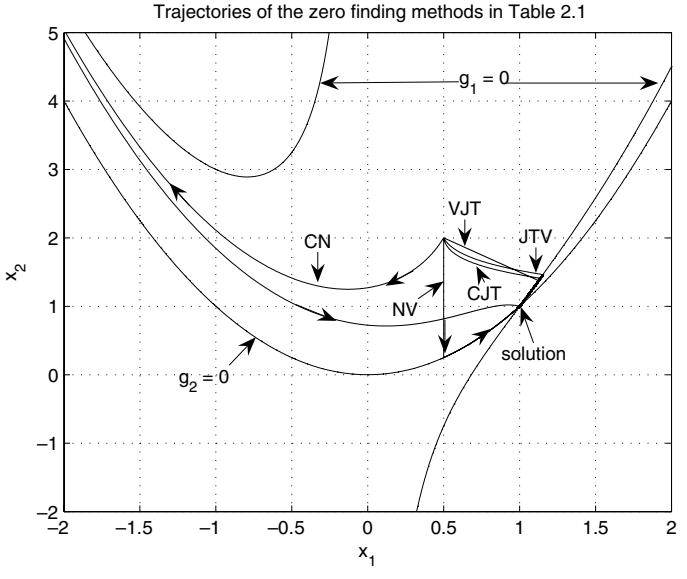

Figure 2.5. *Comparison of trajectories of the zero finding dynamical systems of Table* 2.1 *for minimization of Rosenbrock's function* (2.44)*, with* $a = 0.5$*,* $b = 1$*, or equivalently, finding the zeros of* $\mathbf{g}$ *in* (2.45)*.*

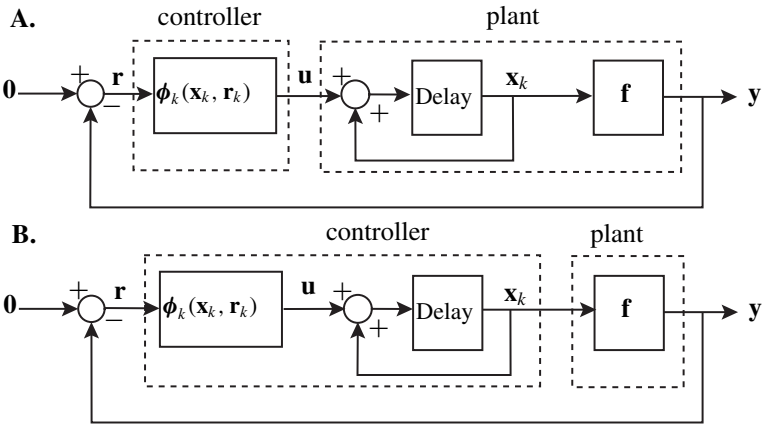

Figure 2.6. A: *A discrete-time dynamical system realization of a general iterative method represented as a feedback control system. The plant, object of the control, represents the problem to be solved, while the controller is a representation of the algorithm designed to solve it. As quadruples, plant* $\mathcal{P} = \{\mathbf{I}, \mathbf{I}, \mathbf{f}, \mathbf{0}\}$ *and controller* $\mathcal{C} = \{\mathbf{0}, \mathbf{0}, \mathbf{0}, \boldsymbol{\phi}_k(\mathbf{x}_k, \mathbf{r}_k)\}$. B: *An alternative discrete-time dynamical system realization of a general iterative method, represented as a feedback control system. As quadruples,* $\mathcal{P} = \{\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{f}\}$ *and* $\mathcal{C} = \{\mathbf{I}, \boldsymbol{\phi}_k(\mathbf{x}_k, \mathbf{r}_k), \mathbf{I}, \mathbf{0}\}$.

Now assume that we define an iteration in $\mathbf{x}$ as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{u}_k, \tag{2.59}$$

where $\mathbf{u}_k$ is a control to be chosen. From (2.59):

$$\mathbf{f}(\mathbf{x}_{k+1}) = \mathbf{f}(\mathbf{x}_k + \mathbf{u}_k),$$

and, using (2.58) and the Taylor expansion of the right-hand side around $\mathbf{x}_k$ and keeping only the first-order term, yields

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \mathbf{D}_\mathbf{f}\mathbf{u}_k, \tag{2.60}$$

where $\mathbf{D}_\mathbf{f} := \mathbf{D}_\mathbf{f}(\mathbf{x}_k)$ is the Jacobian matrix of $\mathbf{f}$. Note that (2.60) defines $\mathbf{r}_{k+1}$ and our task is to choose $\mathbf{u}_k$ using the CLF/LOC approach in order that $\mathbf{r}_k = -\mathbf{f}(\mathbf{x}_k) \to \mathbf{0}$ as $k \to \infty$.

Note that an equivalent interpretation of (2.60) is that it is the forward Euler discretization of (2.13). In this interpretation, it is assumed that the choice of stepsize has been absorbed into the control $\mathbf{u}_k$. If the general expression $\boldsymbol{\phi}_k(\mathbf{x}_k, \mathbf{r}_k)$ defines the term $\mathbf{u}_k$ in (2.59), then the iterative method can be represented as a feedback control system as shown in Figure 2.6.

The quadratic CLF/LOC lemma

Section 2.1 introduced some simple choices of feedback laws, found by a CLF/LOC approach that led to corresponding continuous-time zero finding dynamical systems. A natural

idea is to try and discretize the latter in order to come up with discrete-time dynamical systems (i.e., iterative algorithms in the conventional sense) that determine zeros. As the preceding discussion shows, this leads to (2.60). From this point of view, all that needs to be done is to choose an appropriate time-varying stepsize. In keeping with the philosophy of the previous section, this choice of time-varying stepsize will also be done using the CLF/LOC approach, by postulating that the control can be written as

$$\mathbf{u}_k = \alpha_k \boldsymbol{\phi}(\mathbf{r}_k), \tag{2.61}$$

where α_k represents the time-varying stepsize that is to be chosen, and $\boldsymbol{\phi}(\mathbf{r}_k)$ represents the choice of feedback law. Substituting (2.61) in (2.60) gives

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \boldsymbol{\psi}(\mathbf{r}_k), \tag{2.62}$$

where

$$\boldsymbol{\psi}(\mathbf{r}_k) = \mathbf{D}_\mathbf{f} \boldsymbol{\phi}(\mathbf{r}_k). \tag{2.63}$$

In order to choose α_k, (2.62) will be thought of as a discrete-time dynamical system, affine in the scalar control α_k. The following lemma uses a quadratic CLF and LOC to derive α_k and does not assume the particular form (2.63) so that it can be used repeatedly in what follows, in different situations.

Lemma 2.4 CLF/LOC lemma. *The zero solution of the discrete-time dynamical system* (2.62) *is asymptotically stable if* α_k *is chosen as*

$$\alpha_k = \frac{\langle \mathbf{r}_k, \boldsymbol{\psi}(\mathbf{r}_k) \rangle}{\langle \boldsymbol{\psi}(\mathbf{r}_k), \boldsymbol{\psi}(\mathbf{r}_k) \rangle}, \tag{2.64}$$

provided that

$$\boldsymbol{\psi}(\mathbf{r}_k) \neq \mathbf{0}, \ \textit{whenever } \mathbf{r}_k \neq \mathbf{0}. \tag{2.65}$$

Proof. Consider the quadratic CLF

$$V(\mathbf{r}_k) = \langle \mathbf{r}_k, \mathbf{r}_k \rangle = \mathbf{r}_k^T \mathbf{r}_k = \|\mathbf{r}_k\|^2.$$

Taking the inner product of each side of (2.62) with itself gives

$$\langle \mathbf{r}_{k+1}, \mathbf{r}_{k+1} \rangle = \langle \mathbf{r}_k - \alpha_k \boldsymbol{\psi}(\mathbf{r}_k), \mathbf{r}_k - \alpha_k \boldsymbol{\psi}(\mathbf{r}_k) \rangle,$$

which can be expanded and rearranged as

$$\Delta V := \langle \mathbf{r}_{k+1}, \mathbf{r}_{k+1} \rangle - \langle \mathbf{r}_k, \mathbf{r}_k \rangle = -2\alpha_k \langle \mathbf{r}_k, \boldsymbol{\psi}(\mathbf{r}_k) \rangle + \alpha_k^2 \langle \boldsymbol{\psi}(\mathbf{r}_k), \boldsymbol{\psi}(\mathbf{r}_k) \rangle. \tag{2.66}$$

The LOC choice of α_k is the one that makes ΔV as negative as possible and is found by setting the partial derivative of ΔV with respect to α_k equal to zero.

$$\frac{\partial \Delta V}{\partial \alpha_k} = -2\langle \mathbf{r}_k, \boldsymbol{\psi}(\mathbf{r}_k) \rangle + 2\alpha_k \langle \boldsymbol{\psi}(\mathbf{r}_k), \boldsymbol{\psi}(\mathbf{r}_k) \rangle = 0$$

Table 2.3. *The entries of the first and fourth columns define an iterative zero finding algorithm* $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \boldsymbol{\phi}(\mathbf{r}_k) = \mathbf{x}_k + \alpha_k \boldsymbol{\phi}(-\mathbf{f}(\mathbf{x}_k))$ *(see Theorem* 2.5*). The matrices in the third and fifth rows are defined, respectively, as* $\mathbf{M}_k = \mathbf{D_f P D_f}^T$, $\mathbf{W}_k = \mathbf{D_f D_f}^T$.

$\phi(\mathbf{r}_k)$	$\psi(\mathbf{r}_k)$	Acronym	LOC choice of α_k
$\mathbf{D_f}^{-1}\mathbf{r}_k$	$\mathbf{r}_k$	DN	1
$\mathbf{D_f}^{-1}\mathrm{sgn}(\mathbf{r}_k)$	$\mathrm{sgn}(\mathbf{r}_k)$	DNV	$\dfrac{\Vert\mathbf{r}_k\Vert_1}{\Vert\mathrm{sgn}(\mathbf{r}_k)\Vert_2^2}$
$\mathbf{P D_f}^T\mathbf{r}_k$	$\mathbf{D_f P D_f}^T\mathbf{r}_k$	DJT	$\dfrac{\langle\mathbf{r}_k, \mathbf{M}_k\mathbf{r}_k\rangle}{\Vert\mathbf{M}_k\mathbf{r}_k\Vert_2^2}$
$\mathrm{sgn}(\mathbf{D_f}^T\mathbf{r}_k)$	$\mathbf{D_f}\mathrm{sgn}(\mathbf{D_f}^T\mathbf{r}_k)$	DVJT	$\dfrac{\Vert\mathbf{D_f}^T\mathbf{r}_k\Vert_1}{\Vert\mathbf{D_f}\mathrm{sgn}(\mathbf{D_f}^T\mathbf{r}_k)\Vert_2^2}$
$\mathbf{D_f}^T\mathrm{sgn}(\mathbf{r}_k)$	$\mathbf{D_f D_f}^T\mathrm{sgn}(\mathbf{r}_k)$	DJTV	$\dfrac{\langle\mathbf{r}_k, \mathbf{W}_k\mathrm{sgn}(\mathbf{r}_k)\rangle}{\Vert\mathbf{W}_k\mathrm{sgn}(\mathbf{r}_k)\Vert_2^2}$

yields

$$\alpha_k = \frac{\langle\mathbf{r}_k, \boldsymbol{\psi}(\mathbf{r}_k)\rangle}{\langle\boldsymbol{\psi}(\mathbf{r}_k), \boldsymbol{\psi}(\mathbf{r}_k)\rangle},$$

which leads to

$$\Delta V = -\frac{\langle\mathbf{r}_k, \boldsymbol{\psi}(\mathbf{r}_k)\rangle^2}{\langle\boldsymbol{\psi}(\mathbf{r}_k), \boldsymbol{\psi}(\mathbf{r}_k)\rangle} < 0 \quad \text{for all } \mathbf{r}_k \neq \mathbf{0}, \tag{2.67}$$

since the numerator is a square and the denominator is a squared two-norm. □

From this lemma, the following theorem is immediate.

Theorem 2.5. *Given the function* $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$*, consider the iterative algorithm*

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \boldsymbol{\phi}(-\mathbf{f}(\mathbf{x}_k)), \tag{2.68}$$

where α_k *satisfies* (2.64) *and the function* $\boldsymbol{\phi}(\cdot)$ *is well defined in some neighborhood* $\mathcal{N}(\mathbf{x}^*)$ *of a zero* $\mathbf{x}^*$ *of* $\mathbf{f}(\cdot)$ *and, in addition, satisfies* (2.65)*. Then, for each initial condition* $\mathbf{x}_0 \in \mathcal{N}(\mathbf{x}^*)$*, the corresponding trajectory of* (2.68) *converges to the zero* $\mathbf{x}^*$ *of* $\mathbf{f}(\cdot)$.

It remains to substitute each of the five specific choices of control from section 2.1 (Table 2.1) into the formula (2.64), check (2.65), and calculate the resulting α_k's, to get discrete-time versions, of the form (2.68), of the continuous-time algorithms of section 2.1.

The results correspond to the different iterative methods presented below and tabulated in Table 2.3, which gives the specific LOC choices of stepsize α_k.

The discrete-time *Newton method (DN)*, also called the *Newton–Raphson* method, is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{D_f}^{-1}\mathbf{f}(\mathbf{x}_k). \tag{2.69}$$

The discrete-time *Newton variable structure method (DNV)* is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{D_f}^{-1}\mathrm{sgn}(\mathbf{f}(\mathbf{x}_k)). \tag{2.70}$$

The discrete-time *Jacobian matrix transpose method (DJT)* is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{P}\mathbf{D}_{\mathbf{f}}^T \mathbf{f}(\mathbf{x}_k). \tag{2.71}$$

The discrete-time *variable structure Jacobian matrix transpose method (DVJT)* is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \operatorname{sgn}(\mathbf{D}_{\mathbf{f}}^T \mathbf{f}(\mathbf{x}_k)). \tag{2.72}$$

The discrete-time *Jacobian matrix transpose variable structure method (DJTV)* is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{D}_{\mathbf{f}}^T \operatorname{sgn}(\mathbf{f}(\mathbf{x}_k)). \tag{2.73}$$

Observe that the CLF/LOC lemma can also be used to derive discrete-time versions of the dynamic controller–based continuous-time algorithms in Table 2.2. This derivation is not carried out here (see [PB06]), but is similar to the one in section 2.3.2, in which two coupled discrete-time iterations, which represent a dynamic controller for a linear equation, are analyzed using the CLF/LOC lemma: The discrete-time algorithms corresponding to the dynamic controller-based continuous-time algorithms DC1, DC2, DC3, DC4 in Table 2.2 are given the acronyms DDC1, DDC2, etc.

Comparison of discrete-time methods in Table 2.3

Some numerical examples of the application of the various discrete-time algorithms are given. Once again, these are illustrative examples, and further research needs to be done in order to determine the effectiveness of the new algorithms, beyond the fact that the trajectories take different routes to the zeros to which they converge.

Example 2.6. Branin's example (2.46) is used here to compare the different discrete-time zero finding algorithms proposed in Table 2.3. In Figure 2.7, we show the behavior of the discrete-time algorithms with an LOC choice of stepsize (described in Table 2.3) for this example, with the parameter $c = 0.05$ (one of the values studied in [Bra72, p. 510ff]). Figures 2.9 and 2.10 show the behavior of the static controller–based DJT and DJTV algorithms, compared with the dynamic controller–based algorithms DDC1 and DDC2, with the parameter $c = 0.5$.

Continuous-time algorithms as a first step in the design of discrete-time algorithms

We put the simple but useful quadratic CLF/LOC lemma in perspective by pointing out that, in the discrete-time case, it is no longer as easy to use the 1-norm as it was in the continuous-time case. This is because the requisite manipulations to arrive at a majorization or equivalent expression for the decrement ΔV in the 1 norm are not easy to carry out.

Thus the strategy is to use the continuous-time algorithms to get at the form of the (feedback) controls. Then, thinking of the discrete-time algorithm as a (forward Euler) variable stepsize discretization of the continuous-time algorithm, the stepsize is interpreted as a control, whereas the (possibly nonlinear) feedback law is now interpreted as part of a (possibly nonlinear) system, affine in the control (stepsize). This is where the quadratic CLF/LOC lemma comes in, giving a simple way to choose the variable stepsize, using the familiar quadratic (2-norm) CLF and LOC strategy.

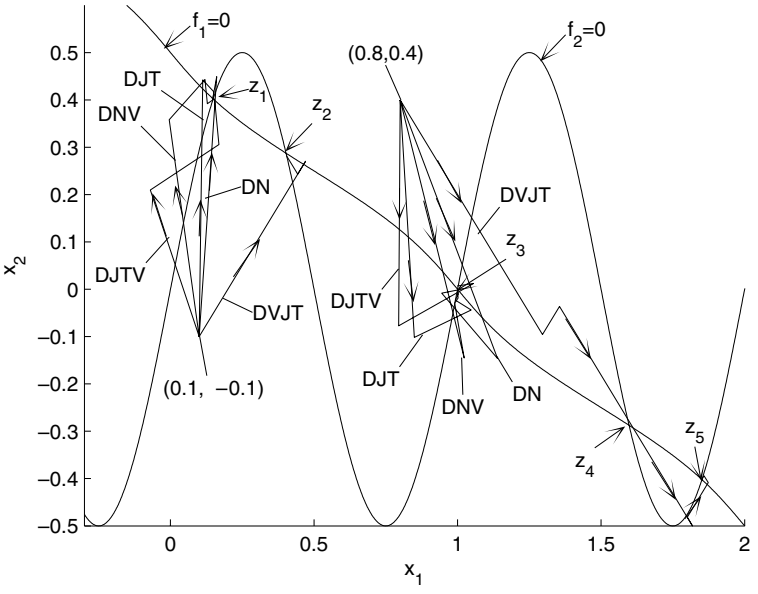

Figure 2.7. *Comparison of different discrete-time methods with LOC choice of stepsize (Table* 2.3*) for Branin's function with* $c = 0.05$*. Calling the zeros* z_1 *through* z_5 *(from left to right), observe that, from initial condition* $(0.1, -0.1)$*, the algorithms DN, DNV, DJT, DJTV converge to* z_1*, whereas DVJT converges to* z_2*. From initial condition* $(0.8, 0.4)$*, once again, DN, DNV, DJT, and DJTV converge to the same zero (*z_3*), whereas DVJT converges to* z_5*.*

Simplifying the adaptive stepsize formulas

From a computational point of view, it is simpler, whenever possible, to avoid the complex adaptation formulas for α_k in the last column of Table 2.3 and to use a simplified stepsize formula.

This is exemplified by taking another closer look at ΔV for the DNV:

$$\Delta V = \langle \mathbf{r}_{k+1}, \mathbf{r}_{k+1} \rangle - \langle \mathbf{r}_k, \mathbf{r}_k \rangle = -2\langle \alpha_k \mathrm{sgn}(\mathbf{r}_k), \mathbf{r}_k \rangle + \langle \alpha_k \mathrm{sgn}(\mathbf{r}_k), \alpha_k \mathrm{sgn}(\mathbf{r}_k) \rangle. \quad (2.74)$$

From (2.74), clearly a sufficient condition for $\Delta V < 0$ is

$$\alpha_k \leq (2\|\mathbf{r}_k\|_1/n). \quad (2.75)$$

Use of (2.75) to choose α_k is not optimal (in the LOC sense), but could lead to some simplification. For example, if $\|\mathbf{r}_k\|_1$ is being monitored (to measure progress to convergence), then it is enough to keep α_k piecewise constant and change its value only when (2.75) is about to be violated.

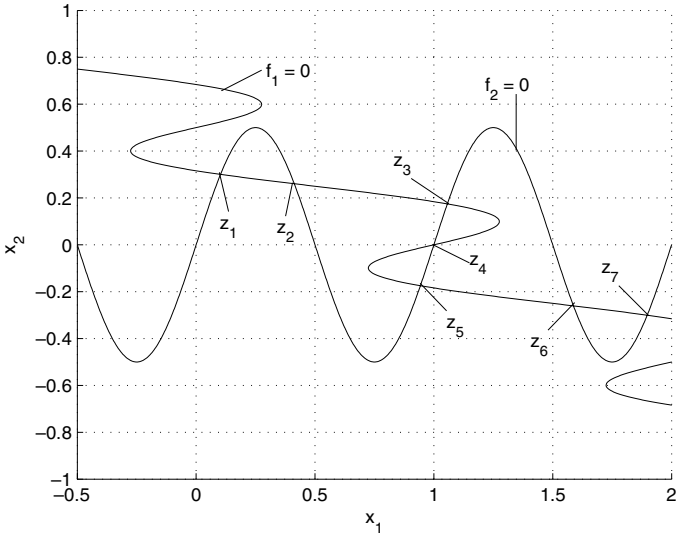

Figure 2.8. *Plot of Branin's function* (2.46)*, with* $c = 0.5$*, showing its seven zeros,* z_1 *through* z_7*. This value of c is used in Figure* 2.9*.*

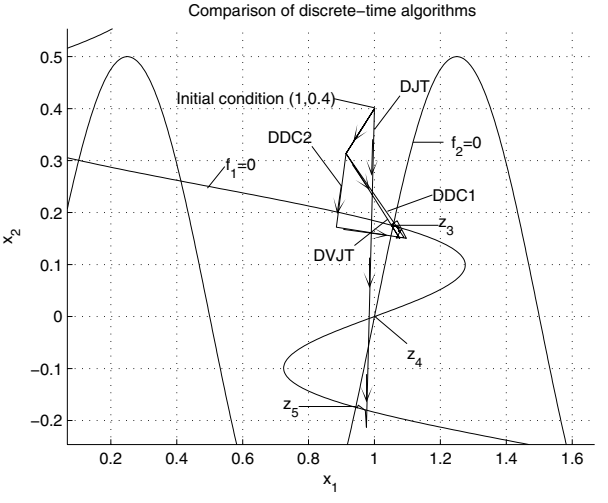

Figure 2.9. *Comparison of the trajectories of the DJT, DVJT, DDC*1*, and DDC*2 *algorithms from the initial condition* $(1, 0.4)$*, showing convergence to the zero* z_5 *of the DJT algorithm and to the zero* z_3 *of the other three algorithms. Note that this initial condition is outside the basin of attraction of the zero* z_3 *for all the other algorithms, including the Newton algorithm (DN). This figure is a zoom of the region around the zeros* z_3*,* z_4*, and* z_5 *in Figure* 2.8*. A further zoom is shown in Figure* 2.10*.*

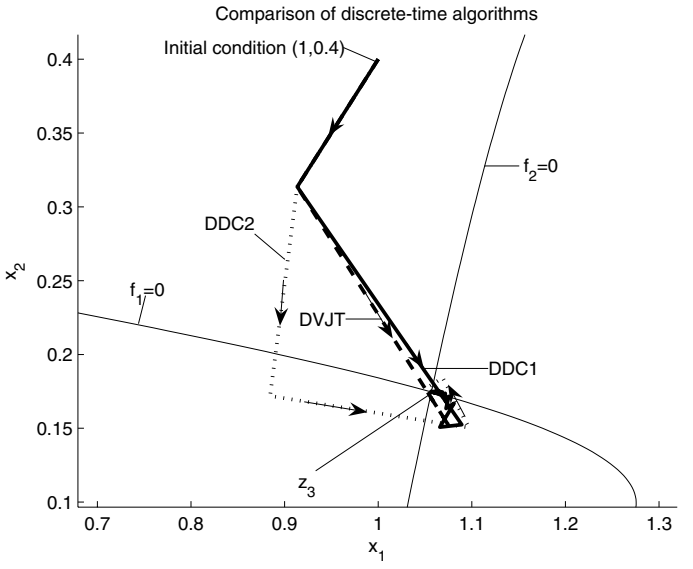

Figure 2.10. *Comparison of the trajectories of the DVJT, DDC*1*, and DDC*2 *algorithms from the initial condition* (1, 0.4)*, showing the paths to convergence to the zero* z_3*. This figure is a further zoom of the region around the zero* z_3 *in Figure* 2.9*.*

"Paradox" of one-step convergence of the discrete Newton method

From the expression for ΔV, it is clear that the choice $\alpha = 1$ is optimal as shown in the first row of Table 2.3, since it maximizes the decrease in the norm of $\mathbf{r}_k$. Indeed, rewriting ΔV as $V(\mathbf{r}_{k+1}) - V(\mathbf{r}_k) = (\alpha^2 - 2\alpha)V(\mathbf{r}_k)$, it appears that, for the LOC choice $\alpha = 1$, $V(\mathbf{r}_{k+1}) = 0$, implying that $\mathbf{r}_{k+1} = \mathbf{0}$. This, of course, is true only for the residue $\mathbf{r}$ in (2.60), which is a first-order approximation or linearization. The real residue, corresponding to the nonlinear Newton iteration (2.68), is given by

$$\mathbf{r}_{k+1} = -\mathbf{f}(\mathbf{x}_{k+1}) = -\mathbf{f}(\mathbf{x}_k - \mathbf{D}_\mathbf{f}^{-1}\mathbf{f}(\mathbf{x}_k)),$$

which is zero only up to first order because

$$\mathbf{f}(\mathbf{x}_k - \mathbf{D}_\mathbf{f}^{-1}\mathbf{f}(\mathbf{x}_k)) = \mathbf{f}(\mathbf{x}_k) - \mathbf{D}_\mathbf{f}\mathbf{D}_\mathbf{f}^{-1}\mathbf{f}(\mathbf{x}_k) + \text{h.o.t.} = \mathbf{0} + \text{h.o.t.},$$

where h.o.t. denotes higher order terms.

Should other discretization methods be used?

We have seen that the discretization of (2.19), using the forward Euler method, results in the standard discrete Newton iterative method. This raises the question of applying different approximation methods to the left-hand side of (2.19) in order to get corresponding discrete iterative methods that belong to the class of Newton methods (because they arise from

discretizations of the continuous Newton method (2.19)), but have different convergence properties. Similar remarks apply to the other static controller–based algorithms in Table 2.3, as well as to the dynamic controller–based continuous algorithms in Table 2.2.

Deeper discussion of this point will take us too far afield, so we will refer the reader to Brezinski [Bre01] and earlier papers [Bog71, BD76, IPZ79] for details. Essentially, Brezinski [Bre01] shows that (i) the Euler method applied to (2.7) is "optimal" in the sense that explicit r-stage Runge–Kutta methods of order strictly greater than 1 cannot have a superlinear order of convergence; and (ii) suitable choice of a variable stepsize results in most of the known and popular methods. We have followed this line of reasoning, adopting the unifying view of the stepsize as a control input.

Condition for CLF of a continuous algorithm to work for its discrete version

A general result is available for variable stepsize Euler discretization. Consider the ODE

$$\dot{\mathbf{x}} = -\mathbf{g}(\mathbf{x}), \qquad \mathbf{x}(\mathbf{0}) = \mathbf{x}_0 \tag{2.76}$$

as well as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - t_k \mathbf{g}(\mathbf{x}_k), \qquad \mathbf{x}_0 \text{ given}, \tag{2.77}$$

where $\mathbf{g} : D \subset \mathbb{R}^n \to \mathbb{R}^n$ is continuous and D is an open convex subset of $\mathbb{R}^n$.

Observe that Euler's method applied to (2.76) with a variable stepsize t_k yields (2.77). Since all iterative methods can be expressed in the form (2.77), (2.76) can be considered as the prototype *continuous analog* of (2.77), also referred to as a continuous algorithm; finally, it is often easier to work with (2.76) to obtain qualitative information on its behavior and then to use this to analyze the iterative method (2.77). Also, as Alber [Alb71] pointed out "theorems concerning the convergence of these (continuous) methods and theorems concerning the existence of solutions of equations and of minimum points of functionals are formulated under weaker assumptions than is the case for the analogous discrete processes."

Boggs [Bog76] observed that it is sometimes difficult to find an appropriate Liapunov function, but that it is often easier to find a Liapunov function for the continuous counterpart (2.76) and then use the same function for (2.77). His result and its simple proof are reproduced below.

Theorem 2.7 [Bog76]. *Let V be a Liapunov function for* (2.76) *at $\mathbf{x}^*$. Assume that ∇V is Lipschitz continuous with constant K on D. Suppose that there is a constant c independent of $\mathbf{x}$ such that $\nabla V^T(\mathbf{x})\mathbf{g}(\mathbf{x}) \geq c\|\mathbf{g}(\mathbf{x})\|^2$. Then there are constants $\underline{t}$ and $\bar{t}$ such that V is a Liapunov function for* (2.77) *at $\mathbf{x}^*$ for $t_k \in [\underline{t}, \bar{t}]$. Furthermore, $\bar{t} < 2c/K$.*

Proof. It only needs to be shown that $\Delta V < 0$ is satisfied along the trajectories of (2.77). Observe that

$$\begin{aligned}\Delta V &= V(\mathbf{x}_k - t_k\mathbf{g}(\mathbf{x}_k)) - V(\mathbf{x}_k) \\ &= \{V(\mathbf{x}_k - t_k\mathbf{g}(\mathbf{x}_k)) - V(\mathbf{x}_k) + t_k \nabla V(x)^T(\mathbf{x}_k)\mathbf{g}(\mathbf{x_k})\} \\ &\quad + [V(\mathbf{x}_k) - t_k \nabla V(x)^T(\mathbf{x}_k)\mathbf{g}(\mathbf{x}_k)] - V(\mathbf{x}_k).\end{aligned}$$

By the Lipschitz condition and by [OR70, Thm. 3, 2.12], the term in braces is bounded by $(1/2)Kt_k^2\|\mathbf{g}(\mathbf{x}_k)\|^2$. Therefore,

$$\Delta V \leq -t_k \nabla V(\mathbf{x})^T \mathbf{g}(\mathbf{x}_k) + (1/2)Kt_k^2\|\mathbf{g}(\mathbf{x}_k)\|^2$$

$$\leq [-t_k c + (1/2)Kt_k^2]\|\mathbf{g}(\mathbf{x}_k)\|^2,$$

which is strictly less than zero if $t_k c > (1/2)Kt_k^2$. Choose $\bar{t} < 2c/K$ and $\underline{t}$ such that $0 < \underline{t} < \bar{t} < 2c/K$, and therefore, for $t \in [\underline{t}, \bar{t}]$ the result follows. □

For the case of steepest descent, $\mathbf{g}(\mathbf{x}) = \nabla \mathbf{f}(\mathbf{x})$ and $\frac{\partial V}{\partial x}^T \mathbf{g}(\mathbf{x}) = \|\mathbf{g}(\mathbf{x})\|^2$, so that $c = 1$, and the steplengths are restricted to the interval $[\underline{t}, 2/K]$.

Clearly, the stepsize can be identified with the control input, and Theorem 2.7 is then seen as a result giving sufficient conditions under which a CLF for the continuous-time system (2.76) works for its discrete counterpart (2.77). Note that the control or stepsize (t_k) is restricted to lie in a bounded interval—a situation which is quite common in control as well. Boggs [Bog76] uses Theorem 2.7 to analyze the Ben-Israel iteration for nonlinear least squares problems; thus his analysis may be viewed as another application of the CLF approach.

Scalar iterative methods

Consider the scalar iterative method

$$x_{k+1} = x_k + u_k f(x_k). \tag{2.78}$$

Various well-known choices of u_k can be arrived at by analyzing the residual (linearized) iteration:

$$r_{k+1} = (1 - f'(x_k)u_k)r_k, \tag{2.79}$$

where $f'(x_k) = \mathrm{d}f(x_k)/\mathrm{d}x$ denotes the derivative (in x) of the function $f(\cdot)$, evaluated at x_k.

Some results for scalar iterative methods are presented in Table 2.4. More details on the order of convergence and choices of u_k for higher order methods that work when $f'(x)$ is not invertible (e.g., when f has a multiple zero), such as the Halley and Chebyshev methods, can be found in [Bre01], where u_k is regarded as a nonstationary stepsize for an Euler method (and accordingly denoted as h_k).

Scalar Newton method subject to disturbances

Using the control formulation of the Newton method discussed in this chapter and first published in [BK03], Kashima and coworkers [KAY05] develop an approach to the scalar Newton iteration with disturbances by treating it as a linear system with nonlinear feedback, known in control terminology as a Lur'e system.

We outline this approach here as an illustration of how control methods lead to further insights into the question of robustness to computational errors in zero finding problems.

Table 2.4. *Showing the choices of control u_k in* (2.78) *that lead to the common variants of the Newton method for scalar iterations.*

Choice of u_k	Name of method
$-f'(x_k)^{-1}$	Newton
$\dfrac{-(x_k - x_{k-1})}{[f(x_k) - f(x_{k-1})]}$	Secant
$\dfrac{f(x_k)}{[f(x_k + f(x_k)) - f(x_k)]}$	Steffensen

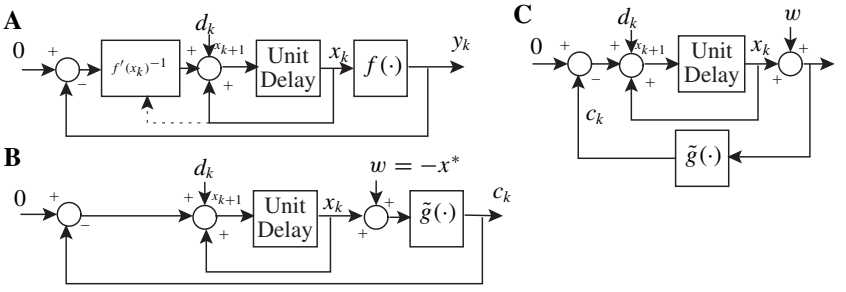

Figure 2.11. *Block diagram manipulations showing how the Newton method with disturbance d_k* (A) *can be redrawn in Lur'e form* (C)*. The intermediate step* (B) *shows the introduction of a constant input w that shifts the input of the nonlinear function $f'^{-1}(\cdot)f(\cdot)$. The shifted function is named $\tilde{g}(\cdot)$. Note that part* A *is identical to Figure* 2.6A*, except for the additional disturbance input d_k.*

Suppose that $f : \mathbb{R} \to \mathbb{R}$ is a continuously differentiable function with nonzero derivative everywhere in $\mathbb{R}$. Assume that there exists x^* such that $f(x^*) = 0$. Then the Newton iteration subject to a disturbance or computational error d_k can be described as follows:

$$x_{k+1} = x_k - f'(x_k)^{-1} f(x_k) + d_k. \tag{2.80}$$

The feedback system description of (2.80) is shown in Figure 2.11A (which is a scalar version of Figure 2.5A, with the addition of a disturbance d_k). Convergence of x_k to x^*, in the absence of computational error ($d_k \equiv 0$), means that the output $y_k = f(x_k)$ converges to the reference input 0. The feedback system of Figure 2.11A may be equivalently redrawn as a system of the Lur'e type, in order to carry out the stability analysis under perturbations. The intermediate steps are explained with reference to Figure 2.11 as follows. In Figure 2.11 B, the function $(f')^{-1} f$ is denoted as g, and the shifted function $g(x + x^*)$ is denoted as $\tilde{g}$. Note that $\tilde{g}(x)$ is 0 for $x = 0$ (i.e., its graph goes through the origin). In addition, in order to use the so-called absolute stability theory, it is assumed that the function $\tilde{g}$ satisfies

a *sector condition*; namely, there exist positive real numbers a and b such that

$$a \leq \frac{\tilde{g}(x)}{x} \leq b \quad \text{for all } x \neq 0. \tag{2.81}$$

This condition is referred to as a sector condition because it means that the graph of the function $\tilde{g}(x)$ is confined between straight lines of positive slope a and b through the origin.

In terms of the new function $\tilde{g}$, the block diagram of Figure 2.11A can be redrawn as in Figure 2.11B, in which the step disturbance w must be chosen as $-x^*$, so that, as before, the output $z_k = \tilde{g}(x_k + w) = \tilde{g}(x_k - x^*) = g(x_k)$ in Figure 2.11A.

This transforms the problem to the classical Lur'e form, since by defining the absolute error as $e_k := x_k - x^*$, the absolute error dynamics is as follows:

$$e_{k+1} = e_k - \tilde{g}(e_k) + d_k. \tag{2.82}$$

In this error system in Lur'e form, the objective is either to guarantee bounded error when disturbances are present or to make the error e_k converge to 0, in the absence of disturbances.

From absolute stability theory [Vid93], the following theorem can be proved.

Theorem 2.8 [KAY05]. *Consider the algorithm* (2.80) *and suppose that there exist constants a and b such that*

$$0 < a \leq \frac{f(x)}{(x - x^*) f'(x)} \leq b < 2 \tag{2.83}$$

for all $x \neq x^*$. Then, for any initial value x_0 and any disturbance sequence $d_k \in \ell^2$, the error sequence e_k also belongs to ℓ^2. Furthermore, if $d_k \equiv 0$, then x_k converges to the exact solution x^*.

Note that this theorem says that, if the disturbance sequence is square-summable, then so is the absolute error sequence.

For a proof of this result, some extensions, and further discussion on the relation of this result to the classical contraction mapping convergence condition, we refer the reader to [KAY05], which is based on the control formulation first given in [BK03].

Region of convergence for Newton's method via Liapunov theory

The Liapunov method can also be used to study the Newton method with disturbances (errors) in the vector case. The first step is to use Corollary 1.18 to find a region of convergence for iterative methods; the Newton method is used as an illustrative example [Hur67, p. 593ff].

Assuming, as usual, that $\mathbf{D_f}(\mathbf{x}_k)$ always has an inverse and that the desired zero $\mathbf{x}^*$ is simple, let the *absolute error* be defined as

$$\mathbf{e}_k := \mathbf{x}_k - \mathbf{x}^*. \tag{2.84}$$

Since $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$, the Taylor expansion of $\mathbf{f}$ around $\mathbf{x}^*$ becomes

$$\mathbf{f}(\mathbf{x}^* + \mathbf{e}_k) = \left[\mathbf{D_f}(\mathbf{x}^*)\right] \mathbf{e}_k + \mathbf{h}(\mathbf{e}_k), \tag{2.85}$$

where $\mathbf{h}(\cdot)$ denotes the higher order terms. Define the matrices

$$\mathbf{M}_1(\mathbf{e}_k) := \left[\mathbf{D_f}(\mathbf{x}^* + \mathbf{e}_k)\right]^{-1}, \quad \mathbf{M}_2(\mathbf{e}_k) := \left[\mathbf{D_f}(\mathbf{x}^* + \mathbf{e}_k) - \mathbf{D_f}(\mathbf{x}^*)\right]. \tag{2.86}$$

Subtracting $\mathbf{x}^*$ from both sides of the Newton iteration

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{D_f}(\mathbf{x}_k)]^{-1}\,\mathbf{f}(\mathbf{x}_k), \tag{2.87}$$

the difference equation for the absolute error can be written as follows:

$$\mathbf{e}_{k+1} = \mathbf{M}_1(\mathbf{e}_k)\,[\mathbf{M}_2(\mathbf{e}_k)\mathbf{e}_k - \mathbf{h}(\mathbf{e}_k)]\,. \tag{2.88}$$

If $\mathbf{x}^*$ is a simple zero of $\mathbf{f}(\cdot)$ and $\mathbf{f}$ is twice continuously differentiable at $\mathbf{x}^*$, then for each $\eta > 0$, there exists a positive constant $c(\eta)$ such that for all $\mathbf{e}$ and for any vector norm $\|\cdot\|$:

$$\|\mathbf{e}\| < \eta \Rightarrow \|\mathbf{M}_1(\mathbf{e})\,[\mathbf{M}_2(\mathbf{e})\mathbf{e} - \mathbf{h}(\mathbf{e})]\,\| \leq c(\eta)\|\mathbf{e}\|^2. \tag{2.89}$$

Choosing the Liapunov function as

$$V(\mathbf{e}) = \|\mathbf{e}\|, \tag{2.90}$$

it follows that

$$\Delta V(\mathbf{e}) \leq -(1 - c(\eta)\|\mathbf{e}\|)\|\mathbf{e}\|, \tag{2.91}$$

which is nonpositive if

$$c(\eta)\|\mathbf{e}\| \leq 1. \tag{2.92}$$

From Corollary 1.18, it follows that a region of convergence for the Newton method is

$$G(\eta_0) := \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}^*\| < \eta_0\}, \tag{2.93}$$

where

$$\eta_0 := \min\{\eta, 1/c(\eta)\}. \tag{2.94}$$

The parameter η can then be chosen so as to maximize η_0, in order to obtain the largest region of convergence corresponding to the choice of Liapunov function (2.90).

Effect of disturbances on the Newton method via Liapunov theory

Corollary 1.20 is used to study the Newton method subject to a disturbance, modeled by the following discrete dynamical system, which is a vector version of (2.80):

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{D_f}(\mathbf{x}_k)]^{-1}\,\mathbf{f}(\mathbf{x}_k) + \mathbf{d}_k, \tag{2.95}$$

where the error term $\mathbf{d}_k$ is only known in terms of an upper bound

$$\|\mathbf{d}_k\| \leq \varepsilon, \tag{2.96}$$

in some norm $\|\cdot\|$, and for some $\varepsilon > 0$. Mimicking the calculations of a convergence region for the Newton method, the absolute error, defined as in (2.84), has the dynamics

$$\mathbf{e}_{k+1} = \mathbf{M}_1(\mathbf{e}_k)\,[\mathbf{M}_2(\mathbf{e}_k)\mathbf{e}_k - \mathbf{h}(\mathbf{e}_k)] + \overline{\mathbf{d}}_k(\mathbf{e}_k), \tag{2.97}$$

where $\mathbf{M}_i$, $i = 1, 2$, are defined as in (2.86). Using the Liapunov function (2.90), the decrement is (cf. (2.91))

$$\Delta V(\mathbf{e}) \leq -(1 - c(\eta)\|\mathbf{e}\|)\|\mathbf{e}\| + \varepsilon =: W(\mathbf{e}) \leq \varepsilon. \tag{2.98}$$

The set S in (1.72) is defined, in this case, as

$$S := \{\mathbf{e} : W(\mathbf{e}) \geq 0\} = \{\mathbf{e} : \|\mathbf{e}\| \leq b\}, \tag{2.99}$$

where

$$b(\eta) = b = \frac{1 - \sqrt{1 - 4c(\eta)\varepsilon}}{2c(\eta)} = \varepsilon + 2c(\eta)\varepsilon^2 + \cdots, \tag{2.100}$$

provided that $4c(\eta)\varepsilon < 1$. If $4c(\eta)\varepsilon \geq 1$, then $W(\mathbf{e}) \geq 0$ everywhere, and the iterations may not converge. Finally, the set A in (1.72) is defined, in this case, as follows:

$$A := \{\mathbf{e} : V(\mathbf{e}) \leq b + \varepsilon\} = \{\mathbf{e} : \|\mathbf{e}\| \leq b + \varepsilon\}. \tag{2.101}$$

Clearly, for η small enough,

$$W(\mathbf{e}) \leq -(b - c(\eta)(b + \varepsilon)^2) = -\delta. \tag{2.102}$$

From Corollary 1.20, if $\delta > 0$, then all solutions that start in $G(\eta_0)$ remain in $G(\eta_0)$, enter A in a finite number of iterations, and remain in A thereafter. The choice of η_0 is now a little more involved than in (2.94); it must be chosen such that

$$c(\eta_0)\eta_0 \leq 1, \quad 4c(\eta_0)\varepsilon < 1, \quad b(\eta_0) - c(\eta_0)(b(\eta_0) + \varepsilon)^2 > 0.$$

If η_1 is chosen as the smallest positive solution of $\eta_1 c(\eta_1) = 1$, then choosing $\eta_0 < \eta_1$ will satisfy, simultaneously, the inequalities $c(\eta_0)\eta_0 < 1$ and $b - c(\eta_0)(b + \varepsilon)^2 > 0$. The remaining condition $4c(\eta_0)\varepsilon < 1$ can then be interpreted as a condition on the precision or accuracy required in the computation, clearly showing that disturbances reduce the region of convergence. If the disturbance is roundoff error, it is more realistic to replace $\mathbf{d}_k$ by $\mathbf{d}_k(\mathbf{x}_k)$, allowing for a roundoff error that depends on the vector $\mathbf{x}_k$, i.e., in control language, a state-dependent disturbance [Hur67]. Note that the analysis above is still valid, provided that the upper bound (2.96) holds for all $\mathbf{x}_k$. In light of this observation, we now refer to the disturbance as roundoff error. Another effect of roundoff errors that can be observed from this Liapunov analysis is that the error in the calculation of each $\mathbf{x}_k$ cannot, in general, be reduced below the value $b + \varepsilon = 2\varepsilon + 2c(\eta)\eta^2 + \cdots$, regardless of the number of iterations carried out. Thus this value is called the *ultimate accuracy* obtainable in the presence of roundoff errors. For small ε, the ultimate accuracy can be seen to be approximately 2ε, which is twice the roundoff error made at each step. The smaller the ultimate accuracy, the better the method is judged to be, in terms of sensitivity to roundoff errors, and, in this respect, the Newton method is a good method.

2.3 Iterative Methods for Linear Systems as Feedback Control Systems

This section specializes the discussion of the previous section, focusing on iterative methods to solve linear systems of equations of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{2.103}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$. First, assuming that $\mathbf{A}$ is nonsingular, (2.103) has a unique solution $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b} \in \mathbb{R}^n$, which it is desired to find, without explicitly inverting the matrix $\mathbf{A}$. A brief discussion of Krylov subspace iterative methods is followed by a control perspective on these and other methods.

Krylov subspace iterative methods

In order to solve the system of linear equations $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is an $n \times n$ nonsingular matrix and $\mathbf{b}$ is a given n-vector, one classical method is Gaussian elimination. It requires, for a dense matrix, storage of all n^2 entries of the matrix and approximately $2n^3/3$ arithmetic operations. Many matrices that arise in practical applications are, however, quite sparse and structured. A typical example is a matrix that arises in the numerical solution of a partial differential equation: Such a matrix has only a few nonzero entries per row. Other applications lead to banded matrices, in which the (i, j)-entry is zero whenever $|i - j| > m$. In general, Gaussian elimination can take only partial advantage of such structure and sparsity.

Matrix-vector multiplication can, on the other hand, take advantage of sparsity and structure. The reason for this is that, if a square $n \times n$ matrix has only k nonzero entries per row ($k \ll n$), then the product of this matrix with a vector will need just kn operations, compared to the $2n^2$ operations that would be required for a general dense matrix-vector multiplication. In addition, only the few nonzero entries of the matrix need to be stored.

The observations made in the preceding paragraphs lead to the question of whether it is possible to solve, at least to a good approximation, a linear system using mainly matrix-vector multiplications and a small number of additional operations. For, if this can be done, the corresponding iterative solution method should certainly be superior to Gaussian elimination, in terms of both computational effort and memory storage requirements.

For iterative methods, an initial guess for the solution is needed to start the iterative method. If no such guess is available, a natural first guess is some multiple of $\mathbf{b}$:

$$\mathbf{x}_1 \in \text{span}\,\{\mathbf{b}\}. \tag{2.104}$$

The next step is to compute the matrix-vector product $\mathbf{Ab}$ and take the next iterate to be some linear combination of the vectors $\mathbf{b}$ and $\mathbf{Ab}$:

$$\mathbf{x}_2 \in \text{span}\,\{\mathbf{b}, \mathbf{Ab}\}. \tag{2.105}$$

Proceeding in this way, at the kth step:

$$\mathbf{x}_k \in \text{span}\,\{\mathbf{b}, \mathbf{Ab}, \ldots, \mathbf{A}^{k-1}\mathbf{b}\}. \tag{2.106}$$

The subspace on the right-hand side of (2.106) is called, in numerical linear algebra, the *Krylov subspace* associated to the pair $(\mathbf{A}, \mathbf{b})$, denoted $\mathcal{K}_k(\mathbf{A}, \mathbf{b})$. In control theory, this subspace is a controllability subspace, as seen in Chapter 1.

Given that $\mathbf{x}_k$ is to be taken from the Krylov subspace $\mathcal{K}_k(\mathbf{A}, \mathbf{b})$, the two main questions that must be answered about such methods can be posed as follows [Gre97]: (i) How good an approximate solution to $\mathbf{Ax} = \mathbf{b}$ is contained in the Krylov subspace? (ii) How can a good (optimal) approximation from this subspace be computed with a moderate amount of work and storage?

Modifying Krylov subspaces by preconditioners

If the Krylov subspace $\mathcal{K}_k(\mathbf{A}, \mathbf{b})$ does not contain a good approximation of the solution for some reasonably small value of k, then a remedy might be to modify the original problem to obtain a better Krlyov subspace. One way to achieve this is to use a so-called preconditioner and solve the modified problem

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{x} = \mathbf{P}^{-1}\mathbf{b} \tag{2.107}$$

by generating approximate solutions $\mathbf{x}_1, \mathbf{x}_2, \ldots$ that satisfy

$$\mathbf{x}_k = \text{Span}\,\{\mathbf{P}^{-1}\mathbf{b}, (\mathbf{P}^{-1}\mathbf{A})\mathbf{P}^{-1}\mathbf{b}, \ldots, (\mathbf{P}^{-1}\mathbf{A})^{k-1}\mathbf{P}^{-1}\mathbf{b}\}. \tag{2.108}$$

Note that at each step of the modified or preconditioned problem, it is necessary to compute the product of $\mathbf{P}^{-1}$ with a vector or, equivalently, to solve a linear system with coefficient matrix $\mathbf{P}$. This means that the matrix $\mathbf{P}$ should be chosen such that the linear system in $\mathbf{P}$ is easy to solve, and specifically, much easier to solve than the original system. The problem of finding a good preconditioner is a difficult one, and although much recent progress has been made, most preconditioners are designed for specific classes of problems, such as those arising in finite element and finite difference approximations of elliptic partial differential equations (PDEs) [Gre97]. In section 5.3, the preconditioning problem is approached from a control viewpoint.

Krylov subspace methods for symmetric matrices

Suppose that an approximation $\mathbf{x}_k$ is called optimal if its residual $\mathbf{b} - \mathbf{A}\mathbf{x}_k$ has minimal Euclidean norm. An algorithm that generates this optimal approximation is called *minimal residual*. If the matrix $\mathbf{A}$ is symmetric, there are known efficient (i.e., short) recurrences to find such an optimal approximation. If, in addition, $\mathbf{A}$ is also positive definite, then another possibility is to minimize the $\mathbf{A}$-norm of the error, $\|\mathbf{e}_k\|_{\mathbf{A}} := \langle \mathbf{A}^{-1}\mathbf{b} - \mathbf{x}_k, \mathbf{b} - \mathbf{A}\mathbf{x}_k\rangle^{1/2}$, and the conjugate gradient algorithm can be shown to do this [Gre97]. Each of these so-called Krylov subspace algorithms carries out one matrix-vector product, as well as a few vector inner products, implying that work and storage requirements are modest. Some Krylov subspace methods are approached from a control viewpoint in sections 2.3.1 and 2.3.2.

Simple iterative methods correspond to static controllers

Following the discussion of the previous section, a general linear iterative method, also called a simple iteration [Gre97], to solve (2.103) can be described by a recurrence of the form

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{K}\mathbf{r}_k, \quad k = 0, 1, 2, \ldots, \tag{2.109}$$

where $\mathbf{K}$ is a real $n \times n$ matrix and the residue $\mathbf{r}_k$, in each iteration, with respect to (2.103), is defined by

$$\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k. \tag{2.110}$$

Exactly as in the previous section, it is possible to associate a discrete-time dynamical feedback system to the iterative method (2.109), and in consequence (2.111) can be viewed as a closed-loop dynamical system with a block diagram representation depicted in Figure

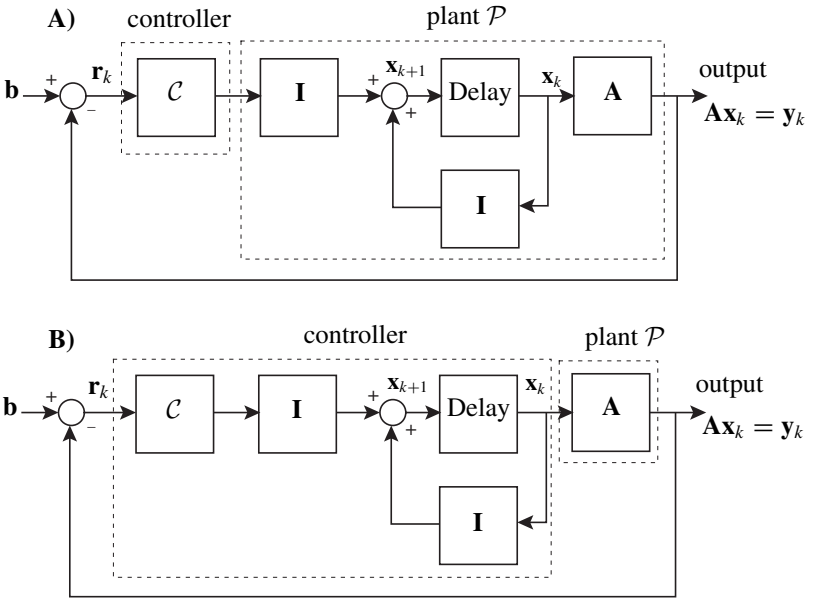

Figure 2.12. A: *A general linear iterative method to solve the linear system of equations* $\mathbf{Ax} = \mathbf{b}$ *represented in standard feedback control configuration. The plant is* $\mathcal{P} = \{\mathbf{I}, \mathbf{I}, \mathbf{A}, \mathbf{0}\}$*, whereas different choices of the linear controller* $\mathcal{C}$ *lead to different linear iterative methods.* B: *A general linear iterative method to solve the linear system of equations* $\mathbf{Ax} = \mathbf{b}$ *represented in an alternative feedback control configuration. The plant is* $\mathcal{P} = \{\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{A}\}$*, whereas different choices of the controller* $\mathcal{C}$ *lead to different iterative methods.*

2.12, where the controller $\mathcal{C}$ is given by $\{\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{K}\}$. The matrix $\mathbf{K}$ is often referred to as the *feedback gain matrix* and we will also use this term whenever appropriate. The observant reader will note that Figure 2.12 is a discrete version with linear plant of Figure 2.1.

Defining $\mathbf{y}_k := \mathbf{Ax}_k$ as the output vector of $S(\mathcal{P}, \mathcal{C})$, consider the constant vector $\mathbf{b}$ as the constant input to this system. The vector $\mathbf{r}_k$ represents the residue or error between the input $\mathbf{b}$ and the output $\mathbf{y}_k$ vectors. The numerical problem of solving the linear system $\mathbf{Ax} = \mathbf{b}$ is thus equivalent to the problem known in control terminology as the *servomechanism* or *regulator* problem of forcing the output $\mathbf{y}$ to regulate to the constant input $\mathbf{b}$, by a suitable choice of controller. When this is achieved, the state vector $\mathbf{x}$ reaches the desired solution of the linear system $\mathbf{Ax} = \mathbf{b}$.

Substituting the expression for $\mathbf{r}_k$ into (2.109), the iterative equation is obtained in the so-called output feedback form, i.e.,

$$\mathbf{x}_{k+1} = (\mathbf{I} - \mathbf{KA})\mathbf{x}_k + \mathbf{Kb}. \tag{2.111}$$

Notice that this corresponds to the choice of a static controller $\mathcal{C} = \{\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{K}\}$ and the iterative method (2.111) corresponds to this particular choice of controller $\mathcal{C}$. We exemplify

this here by the classical *Jacobi iterative method*, described by the recurrence equation

$$\mathbf{x}_{k+1} = \mathbf{H}\mathbf{x}_k + \mathbf{D}^{-1}\mathbf{b}, \tag{2.112}$$

where $\mathbf{H} = \mathbf{D}^{-1}(\mathbf{E} + \mathbf{F})$ and the matrices $\mathbf{D}$, $\mathbf{E}$, and $\mathbf{F}$ are, respectively, strictly diagonal, lower, and upper triangular matrices obtained by splitting matrix $\mathbf{A}$ as $\mathbf{A} = -\mathbf{E} + \mathbf{D} - \mathbf{F}$.

Equating (2.111) and the classical Jacobi iterative equation (2.112), the relationship between the corresponding matrices is given by

$$\mathbf{H} = (\mathbf{I} - \mathbf{K}\mathbf{A}); \quad \mathbf{K} = \mathbf{D}^{-1}. \tag{2.113}$$

Other examples are as follows. If $\mathbf{K} = (\mathbf{D} - \mathbf{E})^{-1}$, then the recurrence (2.111) represents the *Gauss–Seidel* iterative method; if $\mathbf{K} = \left(\omega^{-1}\mathbf{D} - \mathbf{E}\right)^{-1}$, then it represents the *successive overrelaxation (SOR)* method; and, finally, if $\mathbf{K} = \omega\mathbf{D}^{-1}$, then it represents the *extrapolated Jacobi* method. This set of examples should make it clear that all these classical methods correspond to the choice of a static controller $\mathcal{C} = \{\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{K}\}$; the particular choice of $\mathbf{K}$ distinguishes one method from another. The formulation of iterative methods for linear systems as feedback control systems presented here was initiated in [SK01], where shooting methods for ODEs were also analyzed from this perspective. In order to complete the analysis, observe that, in all the cases considered above, the evolution of the residue $\mathbf{r}_k$ is given by the linear recurrence equation below, derived from (2.109) by multiplying both sides by $\mathbf{A}$ and subtracting each side from the vector $\mathbf{b}$:

$$\mathbf{r}_{k+1} = (\mathbf{I} - \mathbf{A}\mathbf{K})\mathbf{r}_k. \tag{2.114}$$

From (2.114) it is clear that convergence of the linear iterative method is ensured if the matrix $\mathbf{S}$ has all its eigenvalues within the unit disk (i.e., is Schur stable), where

$$\mathbf{S} = (\mathbf{I} - \mathbf{A}\mathbf{K}). \tag{2.115}$$

The eigenvalues of this matrix are given by the roots of the characteristic equation:

$$\det(\mathbf{I}\lambda - \mathbf{I} + \mathbf{A}\mathbf{K}) = 0. \tag{2.116}$$

Thus, designing an iterative method with an adequate rate of convergence corresponds to a certain choice of the feedback gain matrix $\mathbf{K}$. This is an instance of the well-studied *inverse eigenvalue problem* known in control as the *problem of pole assignment by state feedback* [Kai80, Son98]. More precisely, (2.114) can be viewed as the dynamical system $\{\mathbf{I}, \mathbf{A}, \mathbf{0}, \mathbf{0}\}$ subject to state feedback with gain matrix $\mathbf{K}$. From standard control theory, it is well known that there exists a state feedback gain $\mathbf{K}$ that results in arbitrary placement of the eigenvalues of the "closed-loop" matrix $\mathbf{S} = \mathbf{I} - \mathbf{A}\mathbf{K}$ if and only if the pair $\{\mathbf{I}, \mathbf{A}\}$ of the quadruple $\{\mathbf{I}, \mathbf{A}, \mathbf{0}, \mathbf{0}\}$ is controllable. Furthermore, the latter occurs if the rank of the controllability matrix is equal to n; i.e.,

$$\text{rank } \mathbf{C} := \text{rank } \left[\mathbf{A} \;\; \mathbf{I}\mathbf{A} \;\; \mathbf{I}^2\mathbf{A} \cdots \mathbf{I}^{n-1}\mathbf{A}\right] = n. \tag{2.117}$$

This condition reduces to rank $\mathbf{A} = n$, i.e., $\mathbf{A}$ nonsingular. We thus have the following lemma.

Lemma 2.9. *Consider the matrices* $\mathbf{A}$ *and* $\mathbf{K}$ *both real and square of dimension* n*. The eigenvalues of the matrix* $\mathbf{S} := (\mathbf{I} - \mathbf{AK})$ *can be arbitrarily assigned by choice of* $\mathbf{K}$ *if and only if the matrix* $\mathbf{A}$ *is nonsingular.*

Actually, it is possible to state a slightly more general form of this lemma, showing that the less stringent requirement of stabilizability also implies that the matrix $\mathbf{A}$ must be nonsingular.

Lemma 2.10. *There exists a matrix* $\mathbf{K}$ *such that* $\rho(\mathbf{S}) = \rho(\mathbf{I} - \mathbf{AK}) < 1$ *if and only if the matrix* $\mathbf{A}$ *is nonsingular.*

Proof. ("If") Choose $\mathbf{K} = \mathbf{A}^{-1}$.

("Only if") The contrapositive is proved. Note first that if $\mathbf{A}$ is singular, then for all matrices $\mathbf{K}$, the product $\mathbf{AK}$ is also singular, and moreover, rank $\mathbf{AK} \leq$ rank $\mathbf{A}$. It now suffices to observe that a singular matrix $\mathbf{Z} \in \mathbb{R}^{n\times n}$ with rank $\mathbf{Z} = p$ has $n - p$ eigenvalues equal to zero. Thus the matrix $\mathbf{I} - \mathbf{Z}$ has $n - p$ eigenvalues equal to 1, and hence $\rho(\mathbf{I} - \mathbf{Z}) \geq 1$. Take $\mathbf{Z} = \mathbf{AK}$. □

Notice that the particular choice $\mathbf{K} = \mathbf{A}^{-1}$ makes all the eigenvalues of matrix $\mathbf{S}$ equal to zero, implying that the iterative scheme (2.114) will converge in one iteration. This is, of course, only a theoretical remark, since if the inverse of matrix $\mathbf{A}$ were in fact available, it would be enough to compute $\mathbf{A}^{-1}\mathbf{b}$ in order to solve the linear system $\mathbf{Ax} = \mathbf{b}$ and unnecessary to resort to any iteration. In fact, the problem of solving a linear system $\mathbf{Ax} = \mathbf{b}$ without inverting $\mathbf{A}$ can be stated in control terms as that of "emulating" $\mathbf{A}^{-1}$ without actually computing it, and this is exactly what iterative methods do. We also remark that the convergence in one iteration, or more generally in a finite number of iterations, is just a question of making the iteration matrix in (2.114) *nilpotent*, with the index of nilpotency representing an upper bound on the number of iterations required to zero the residue. This is another topic, called *deadbeat control*, that is well studied in the control literature in [ÅW84], to which we refer the reader.

Lemma 2.10 says that stabilizability of the pair $\{\mathbf{I}, \mathbf{A}\}$ implies that the matrix $\mathbf{A}$ must be nonsingular. Another result of this nature is that controllability of the pair $\{\mathbf{A}, \mathbf{b}\}$ implies that the system $\mathbf{Ax} = \mathbf{b}$ possesses a unique solution [dSB81]. Actually, there are deeper connections with Krylov subspaces that we will not dwell on here; however, see [IM98].

2.3.1 CLF/LOC derivation of minimal residual and Krylov subspace methods

The next natural question is whether it is possible to do better with more complicated controllers than the static constant controllers of the preceding discussion.

First, consider the case in which matrix $\mathbf{K}$ is no longer a constant and is, in fact, dependent on the state $\mathbf{x}$ or the iteration counter k:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{K}_k\mathbf{r}_k. \tag{2.118}$$

In particular, in many iterative methods, the matrix $\mathbf{K}_k$ is chosen as $\alpha_k \mathbf{I}$, leading to

$$\mathbf{x}_{k+1} = (\mathbf{I} - \alpha_k \mathbf{A})\mathbf{x}_k + \alpha_k \mathbf{b}, \tag{2.119}$$

where α_k is a scalar sequence and $\mathbf{I}$ is an identity matrix of appropriate dimension. One method differs from another in the way in which the scalars α_k are chosen; e.g., if the α_k's are precomputed (from arguments involving clustering of eigenvalues of the iteration matrix), we get the class of Chebyshev-type "semi-iterative" methods; if the α_k are computed in terms of the current values of $\mathbf{r}_k$, the resulting class is referred to as adaptive Richardson, etc.)

The objective of this section is to show that this and some related classes of methods have a natural control interpretation that permits the analysis and design of this class, using the CLF/LOC approach.

Considering the matrix $\mathbf{K}_k$ in (2.118) given by $\mathbf{K}_k = \alpha_k \mathbf{I}$, it is convenient to rewrite (2.118) in terms of the residue $\mathbf{r}_k$ as follows:

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{r}_k. \tag{2.120}$$

In control language, now thinking of the parameter α_k as a control u_k and vector $\mathbf{r}_k$ as the state, (2.120) describes a dynamical system of the type known as *bilinear*, since it is linear in the state $\mathbf{r}$ if the control α is fixed and linear in the control α if the state is fixed. It is not linear if both state and control are allowed to vary, since the right-hand side contains a product term $\alpha_k \mathbf{A}\mathbf{r}_k$ involving the control input and the state $\mathbf{r}_k$.

Since the system is no longer linear or time invariant, straightforward eigenvalue analysis is no longer applicable. A control Liapunov function is used to design an asymptotically stabilizing state feedback control for (2.120) that drives $\mathbf{r}_k$ to the origin and thus solves the original problem (2.103).

Consider the control Liapunov function candidate $V(\mathbf{r}_k) := \langle \mathbf{r}_k, \mathbf{r}_k \rangle = \mathbf{r}_k^T \mathbf{r}_k$. Then, from (2.120),

$$\langle \mathbf{r}_{k+1}, \mathbf{r}_{k+1} \rangle = \langle \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{r}_k, \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{r}_k \rangle \tag{2.121}$$

$$= \langle \mathbf{r}_k, \mathbf{r}_k \rangle - 2\alpha_k \langle \mathbf{r}_k, \mathbf{A}\mathbf{r}_k \rangle + \alpha_k^2 \langle \mathbf{A}\mathbf{r}_k, \mathbf{A}\mathbf{r}_k \rangle, \tag{2.122}$$

from which it follows that

$$\Delta V := V(\mathbf{r}_{k+1}) - V(\mathbf{r}_k) = -\alpha_k \left(2\langle \mathbf{r}_k, \mathbf{A}\mathbf{r}_k \rangle - \alpha_k \langle \mathbf{A}\mathbf{r}_k, \mathbf{A}\mathbf{r}_k \rangle\right). \tag{2.123}$$

From this expression it is clear that the LOC choice

$$\alpha_k = \frac{\langle \mathbf{r}_k, \mathbf{A}\mathbf{r}_k \rangle}{\langle \mathbf{A}\mathbf{r}_k, \mathbf{A}\mathbf{r}_k \rangle} \tag{2.124}$$

leads to

$$\Delta V = -\frac{\langle \mathbf{r}_k, \mathbf{A}\mathbf{r}_k \rangle^2}{\langle \mathbf{A}\mathbf{r}_k, \mathbf{A}\mathbf{r}_k \rangle} < 0, \tag{2.125}$$

showing that the candidate control Liapunov function works and that (2.124) is the appropriate choice of feedback control. Furthermore, ΔV is strictly negative unless $\langle \mathbf{r}_k, \mathbf{A}\mathbf{r}_k \rangle = 0$. One way of saying that this possibility is excluded is to say that zero does not belong to the

field of values of $\mathbf{A}$ [Gre97]. In other words, the control Liapunov function proves that the residual vector $\mathbf{r}_k$ decreases monotonically to the zero vector.

Note that the stabilizing feedback control α_k is a nonlinear function of the state, which is not surprising, since the system being stabilized is not linear, but bilinear. The choice (2.124) results in the so-called Orthomin(1) method [Gre97].

Richardson's method is an LOC/CLF steepest descent method

The observant reader will note that (2.124) corresponds to the LOC choice for the quadratic CLF V and that it could have been derived by the application of Lemma 2.4. Note, however, that it is not necessary to stick to the quadratic CLF $\mathbf{r}^T\mathbf{r}$ in order to use the LOC approach. In particular, a small change in the candidate Liapunov function, together with the assumption that the matrix $\mathbf{A}$ is positive definite, leads to another well-known method. Since $\mathbf{A}$ is positive definite, $\mathbf{A}^{-1}$ exists and the following choice is legitimate:

$$V(\mathbf{r}_k) := \langle \mathbf{r}_k, \mathbf{A}^{-1}\mathbf{r}_k \rangle. \tag{2.126}$$

Repeating the steps above, it is easy to arrive at

$$\Delta V = -\alpha_k \left(2\langle \mathbf{A}\mathbf{r}_k, \mathbf{A}^{-1}\mathbf{r}_k \rangle - \alpha_k \langle \mathbf{A}\mathbf{r}_k, \mathbf{A}^{-1}\mathbf{A}\mathbf{r}_k \rangle \right), \tag{2.127}$$

from which it follows, in exact analogy to the development above, that

$$\alpha_k = \frac{\langle \mathbf{A}\mathbf{r}_k, \mathbf{A}^{-1}\mathbf{r}_k \rangle}{\langle \mathbf{A}\mathbf{r}_k, \mathbf{A}^{-1}\mathbf{A}\mathbf{r}_k \rangle} = \frac{\langle \mathbf{r}_k, \mathbf{r}_k \rangle}{\langle \mathbf{r}_k, \mathbf{A}\mathbf{r}_k \rangle} \tag{2.128}$$

is the appropriate choice of feedback control that makes $\Delta V < 0$, which, in fact, corresponds to *Richardson's iterative method* for symmetric matrices [You89, Var00, SvdV00], sometimes also qualified with the adjectives *adaptive* and *parameter-free*, since the α_k's are calculated in feedback form.

From (2.118), Richardson's method can be written as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{r}_k, \quad \text{where } \alpha_k = \frac{\langle \mathbf{r}_k, \mathbf{r}_k \rangle}{\langle \mathbf{r}_k, \mathbf{A}\mathbf{r}_k \rangle}. \tag{2.129}$$

Now observe that the problem of solving the linear system is equivalent to that of minimizing the quadratic form $\langle \mathbf{x}, \mathbf{A}\mathbf{x} \rangle - 2\langle \mathbf{b}, \mathbf{x} \rangle$ (since the latter attains its minimum where $\mathbf{A}\mathbf{x} = \mathbf{b}$).

Since the negative gradient of this function at $\mathbf{x} = \mathbf{x}_k$ is $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$, clearly (2.129) can be viewed as a *steepest descent method* in which the stepsize α_k is chosen optimally in the LOC/CLF sense. From a control viewpoint, since the control action on the state is $\alpha_k \mathbf{r}_k$, i.e., proportional to the error or residue $\mathbf{r}_k$, Richardson's method can be viewed as the application of a proportional controller with a time-varying gain α_k.

2.3.2 The conjugate gradient method derived from a proportional-derivative controller

In a survey of the top ten algorithms of the century, Krylov subspace methods have a prominent place [DS00, vdV00]. Furthermore, as Trefethen says in his essay on numerical

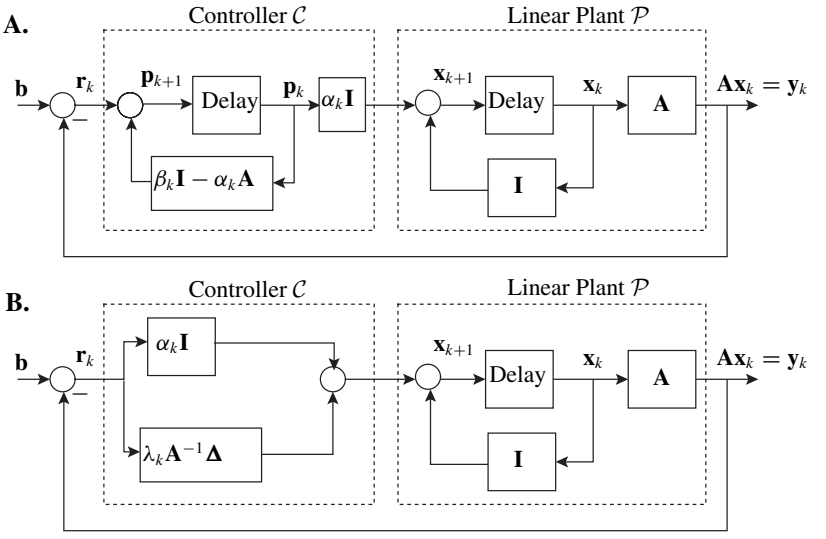

Figure 2.13. A: *The conjugate gradient method represented as the standard plant* $\mathcal{P} = \{\mathbf{I}, \mathbf{I}, \mathbf{A}, \mathbf{0}\}$ *with dynamic nonstationary controller* $\mathcal{C} = \{(\beta_k\mathbf{I} - \alpha_k\mathbf{A}), \mathbf{I}, \alpha_k\mathbf{I}, \mathbf{0}\}$ *in the variables* $\mathbf{p}_k$, $\mathbf{x}_k$. B: *The conjugate gradient method represented as the standard plant* $\mathcal{P}$ *with a nonstationary (time-varying) proportional-derivative (PD) controller in the variables* $\mathbf{r}_k$, $\mathbf{x}_k$, *where* $\lambda_k = \beta_{k-1}\alpha_k/\alpha_{k-1}$. *This block diagram emphasizes the conceptual proportional-derivative structure of the controller. Of course, the calculations represented by the derivative block,* $\lambda_k\mathbf{A}^{-1}\mathbf{\Delta}$, *are carried out using formulas* (2.143), (2.148) *that do not involve inversion of the matrix* $\mathbf{A}$.

analysis: "For guidance to the future we should study not Gaussian elimination and its beguiling stability properties, but the diabolically fast conjugate gradient iteration" [Tre92]. This section shows that the formal conjugate gradient method, one of the best known Krylov subspace methods, is also easily arrived at from a control viewpoint. This has the merit of providing a natural control motivation for the conjugate gradient method in addition to providing some insights as to why it has certain desirable properties, such as speed and robustness, in the face of certain types of errors.

In fact, the conjugate gradient method is conveniently viewed as an acceleration of Richardson's steepest descent method (2.129).

Note that the latter can be viewed as the standard feedback control system $S(\mathcal{P}, \mathcal{C})$ with the controller $\{\mathbf{0}, \mathbf{0}, \mathbf{0}, \alpha_k(\mathbf{r}_k)\mathbf{I}\}$, which is referred to as a *proportional controller*. The acceleration is achieved by using a discrete version of a classical control strategy for faster "closed-loop" response (i.e., acceleration of convergence to the solution). This strategy is known as *derivative action* in the controller. The development of this approach is as follows.

Suppose that a new method is to be derived from the steepest descent method by adding a new term that is proportional to a discrete derivative of the state vector $\mathbf{x}_k$. In other words, the new increment $\Delta\mathbf{x}_k := \mathbf{x}_{k+1} - \mathbf{x}_k$ is a linear combination of the steepest descent direction $\mathbf{r}_k$ and the previous increment or discrete derivative of the state $\mathbf{x}_k - \mathbf{x}_{k-1}$.

Denoting the scalar gains as α_k and γ_k, the new method, depcited in Figure 2.13, can be expressed mathematically as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k[\mathbf{r}_k + \gamma_k(\mathbf{x}_k - \mathbf{x}_{k-1})], \tag{2.130}$$

which can be rewritten as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k, \tag{2.131}$$

where

$$\begin{aligned} \mathbf{p}_k &= \mathbf{r}_k + \gamma_k(\mathbf{x}_k - \mathbf{x}_{k-1}) = \mathbf{r}_k + \gamma_k \alpha_{k-1}\mathbf{p}_{k-1} \\ &= \mathbf{r}_k + \beta_{k-1}\mathbf{p}_{k-1}, \end{aligned} \tag{2.132}$$

and

$$\beta_{k-1} := \gamma_k \alpha_{k-1}. \tag{2.133}$$

Combining these equations leads to

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k, \tag{2.134}$$

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k, \tag{2.135}$$

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k. \tag{2.136}$$

From a control viewpoint, it is natural to think of the "parameters" α_k and β_k as scalar control inputs. The motivation for doing this is the observation that the systems to be controlled then belong to the class of systems known as bilinear, similarly to system (2.120). More precisely, taking $\mathbf{r}_k$ and $\mathbf{p}_k$ as the state variables, it is necessary to analyze the following pair of coupled bilinear systems:

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k, \tag{2.137}$$

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k. \tag{2.138}$$

Provided that α_k is not identically zero, it is easy to see that the equilibrium solution of this system is the zero solution $\mathbf{r}_k = \mathbf{p}_k = \mathbf{0}$ for all k. The control objective is to choose the scalar controls α_k, β_k so as to drive the state vectors $\mathbf{r}_k$ and $\mathbf{p}_k$ to zero. The analysis will be carried out in terms of the variables $\mathbf{r}_k$ and $\mathbf{p}_k$. Thus the objective is to show that the same control Liapunov function approach that has been successfully applied to other iterative methods above can also be used here to derive choices of α_k and β_k that result in stability of the zero solution. The analysis proceeds in two stages. In the first stage, a choice of α_k guided by a control Liapunov function is shown to result in a decrease of a suitable norm of $\mathbf{r}_k$. In the second stage, a second control Liapunov function orients the choice of β_k that results in a decrease of a suitable norm of $\mathbf{p}_k$. The conclusion is that $\mathbf{r}_k$ and $\mathbf{p}_k$ both converge to zero, as required.

Since $\mathbf{A}$ is a real positive definite matrix, so is $\mathbf{A}^{-1}$ and both matrices define weighted 2-norms. The control Liapunov method is used to choose the controls, using the $\mathbf{A}^{-1}$-norm for (2.137) and the $\mathbf{A}$-norm for (2.138). Before proceeding, it should be pointed out that these choices are arbitrary and that exactly the same control Liapunov argument with different choices of norms leads to different methods.

Thus the first step is to calculate the $\mathbf{A}^{-1}$-norm of both sides of (2.137) in order to choose a control α_k that will result in the reduction of this norm of $\mathbf{r}$ to zero:

$$\langle \mathbf{r}_{k+1}, \mathbf{A}^{-1}\mathbf{r}_{k+1}\rangle = \langle \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k, \mathbf{A}^{-1}(\mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k)\rangle \tag{2.139}$$

$$= \langle \mathbf{r}_k, \mathbf{A}^{-1}\mathbf{r}_k\rangle - 2\alpha_k\langle \mathbf{r}_k, \mathbf{p}_k\rangle + \alpha_k^2\langle \mathbf{A}\mathbf{p}_k, \mathbf{p}_k\rangle. \tag{2.140}$$

It follows that

$$\Delta V_{\mathbf{r}} := \langle \mathbf{r}_{k+1}, \mathbf{A}^{-1}\mathbf{r}_{k+1}\rangle - \langle \mathbf{r}_k, \mathbf{A}^{-1}\mathbf{r}_k\rangle = -2\alpha_k\langle \mathbf{r}_k, \mathbf{p}_k\rangle + \alpha_k^2\langle \mathbf{A}\mathbf{p}_k, \mathbf{p}_k\rangle. \tag{2.141}$$

The LOC choice of α_k is found from the calculation

$$\frac{\partial \Delta V}{\partial \alpha_k} = -2\langle \mathbf{r}_k, \mathbf{p}_k\rangle + 2\alpha_k\langle \mathbf{A}\mathbf{p}_k, \mathbf{p}_k\rangle \tag{2.142}$$

so that $\frac{\partial \Delta V}{\partial \alpha_k} = 0$ when

$$\alpha_k = \frac{\langle \mathbf{r}_k, \mathbf{p}_k\rangle}{\langle \mathbf{A}\mathbf{p}_k, \mathbf{p}_k\rangle}. \tag{2.143}$$

This choice of α_k is the LOC choice, i.e., optimal in the sense that it makes ΔV as negative as possible. In other words, it makes the reduction in the $\mathbf{A}^{-1}$-norm of $\mathbf{r}$ as large as possible:

$$\Delta V = -\frac{\langle \mathbf{r}_k, \mathbf{p}_k\rangle^2}{\langle \mathbf{A}\mathbf{p}_k, \mathbf{p}_k\rangle}. \tag{2.144}$$

This derivation of α_k also gives a clue to the robustness of the conjugate gradient algorithm, since the argument so far has not used any information on the properties of the vectors $\mathbf{p}_k$ (such as $\mathbf{A}$-orthogonality). This indicates that, in a finite precision implementation, even when properties such as $\mathbf{A}$-orthogonality are lost, the choice of α_k in (2.143) ensures that the $\mathbf{A}^{-1}$-norm of $\mathbf{r}$ will decrease.

Proceeding with the analysis, consider the "$\mathbf{p}_k$-subsystem" subject to the control β_k. The $\mathbf{A}$-norm of both sides of (2.138) is calculated in order to choose an appropriate control input β_k:

$$\langle \mathbf{p}_{k+1}, \mathbf{A}\mathbf{p}_{k+1}\rangle = \langle \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k, \mathbf{A}(\mathbf{r}_{k+1} + \beta_k \mathbf{p}_k)\rangle \tag{2.145}$$

$$= \langle \mathbf{r}_{k+1}, \mathbf{A}\mathbf{r}_{k+1}\rangle + 2\beta_k\langle \mathbf{p}_k, \mathbf{A}\mathbf{r}_{k+1}\rangle + \beta_k^2\langle \mathbf{p}_k, \mathbf{A}\mathbf{p}_k\rangle. \tag{2.146}$$

Using the same line of argument as above, calculate

$$\frac{\partial \|\mathbf{p}_{k+1}\|_{\mathbf{A}}^2}{\partial \beta_k} = 2\langle \mathbf{p}_k, \mathbf{A}\mathbf{r}_{k+1}\rangle + 2\beta_k\langle \mathbf{p}_k, \mathbf{A}\mathbf{p}_k\rangle, \tag{2.147}$$

so that

$$\beta_k = -\frac{\langle \mathbf{p}_k, \mathbf{A}\mathbf{r}_{k+1}\rangle}{\langle \mathbf{p}_k, \mathbf{A}\mathbf{p}_k\rangle} \tag{2.148}$$

is the LOC choice, resulting in

$$\|\mathbf{p}_{k+1}\|_{\mathbf{A}}^2 = \|\mathbf{r}_{k+1}\|_{\mathbf{A}}^2 - \frac{\langle \mathbf{p}_k, \mathbf{A}\mathbf{r}_{k+1}\rangle^2}{\langle \mathbf{p}_k, \mathbf{A}\mathbf{p}_k\rangle}. \tag{2.149}$$

Since the second term is negative (except at the solution $\mathbf{p}_k = \mathbf{0}$), this results in the inequality

$$\|\mathbf{p}_{k+1}\|_{\mathbf{A}} < \|\mathbf{r}_{k+1}\|_{\mathbf{A}}. \tag{2.150}$$

From (2.144) and the equivalence of norms, it can be concluded that $\mathbf{r}_{k+1}$ decreases in any induced norm (in particular, in the $\mathbf{A}$-norm). Thus (2.150) implies that $\mathbf{p}_{k+1}$ decreases in the $\mathbf{A}$-norm, although not necessarily monotonically.

The above derivations may be summarized in the form of an algorithm, which we will refer to as the CLF/LOC version of the conjugate gradient algorithm.

The conjugate gradient algorithm: CLF/LOC version.
Compute $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{p}_0 := \mathbf{r}_0$.
For $k = 0, 1, \ldots,$ until convergence
Do:
$\alpha_k = \langle \mathbf{r}_k, \mathbf{p}_k \rangle / \langle \mathbf{A}\mathbf{p}_k, \mathbf{p}_k \rangle$
$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$
$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$
$\beta_k = -\langle \mathbf{p}_k, \mathbf{A}\mathbf{r}_{k+1} \rangle / \langle \mathbf{p}_k, \mathbf{A}\mathbf{p}_k \rangle$
$\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
EndDo

Under the assumption that the conjugate gradient method is initialized choosing $\mathbf{r}_0 = \mathbf{p}_0$, (2.143) and (2.148) are equivalent to the more commonly used but less obvious forms [Gre97]

$$\alpha_k = \langle \mathbf{r}_k, \mathbf{r}_k \rangle / \langle \mathbf{p}_k, \mathbf{A}\mathbf{p}_k \rangle, \tag{2.151}$$

$$\beta_k = \langle \mathbf{r}_{k+1}, \mathbf{r}_{k+1} \rangle / \langle \mathbf{r}_k, \mathbf{r}_k \rangle. \tag{2.152}$$

With these choices of α_k and β_k the CLF/LOC version becomes the standard textbook [Saa96, Alg. 6.17, p. 179] version of the conjugate gradient algorithm, given below for comparison.

The standard conjugate gradient algorithm.
Compute $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{p}_0 := \mathbf{r}_0$.
For $k = 0, 1, \ldots,$ until convergence
Do:
$\alpha_k := \langle \mathbf{r}_k, \mathbf{r}_k \rangle / \langle \mathbf{A}\mathbf{p}_k, \mathbf{p}_k \rangle$
$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$
$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$
$\beta_k := \langle \mathbf{r}_{k+1}, \mathbf{r}_{k+1} \rangle / \langle \mathbf{r}_k, \mathbf{r}_k \rangle$
$\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
EndDo

The difference between these two algorithms appears exactly when the assumption $\mathbf{r}_0 = \mathbf{p}_0$ is violated; in this case, it can happen that, for some choices of $\mathbf{r}_0 \neq \mathbf{p}_0$, the standard version of the conjugate gradient algorithm diverges, whereas the CLF/LOC version does not. A practical application in which such a situation occurs is in *adaptive filtering*, which, in the current context, can be described as using an "on-line" conjugate gradient-type algorithm to solve a system of the type $\mathbf{A}_k \mathbf{x}_k = \mathbf{b}_k$. The term "on-line" refers to the updating

of the matrix $\mathbf{A}_k$ and the right-hand side $\mathbf{b}_k$ at each iteration and means, in practice, that the assumption $\mathbf{r}_0 = \mathbf{p}_0$ does not hold after the first iteration. The paper [CW00] discusses the nonconvergent behavior of an "on-line" version of the standard conjugate gradient algorithm in this situation and proposes a solution based on choices of α_k and β_k that involve a heuristic choice of an additional parameter and a line search. In contrast, numerical experiments show that, for this class of problems, an appropriately modified on-line CLF/LOC variant of the conjugate gradient algorithm works well, without the need for line search and heuristically chosen parameters [DB06].

Variants of the conjugate gradient algorithm

The Orthomin(2) algorithm [Gre97] differs from the standard conjugate gradient algorithm only in the choice of the controls α_k and β_k. From the viewpoint adopted here, it can be said that the difference lies in the choice of the norms used for the control Liapunov functions for the $\mathbf{r}$ and $\mathbf{p}$ subsystems. More precisely, consider the algorithm (coupled bilinear systems)

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k, \tag{2.153}$$

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} - \beta_k \mathbf{p}_k. \tag{2.154}$$

Suppose that the 2-norm is used as the control Liapunov function for the $\mathbf{r}$ subsystem and that the 2-norm of $\mathbf{A}\mathbf{p}$ (recall that for the Orthomin(2) method it is not assumed that the matrix $\mathbf{A}$ is symmetric) is the control Liapunov function for the $\mathbf{p}$ subsystem. A calculation that is strictly analogous to the one above for the conjugate gradient method shows that this choice of norms results in

$$\alpha_k = \frac{\langle \mathbf{r}_k, \mathbf{A}\mathbf{p}_k \rangle}{\langle \mathbf{A}\mathbf{p}_k, \mathbf{A}\mathbf{p}_k \rangle}, \quad \beta_k = \frac{\langle \mathbf{A}\mathbf{p}_k, \mathbf{A}\mathbf{r}_{k+1} \rangle}{\langle \mathbf{A}\mathbf{p}_k, \mathbf{A}\mathbf{p}_k \rangle}, \tag{2.155}$$

which is exactly the Orthomin(2) choice of α_k and β_k (see [Gre97]).

The CLF proof of the conjugate gradient choices of α_k, β_k allows another observation that, to the authors' knowledge, has not been made in the literature. Consider the following variant of the conjugate gradient algorithm:

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k, \tag{2.156}$$

$$\mathbf{p}_{k+1} = \mathbf{r}_k + \beta_k \mathbf{p}_k. \tag{2.157}$$

In this version of conjugate gradient, the second equation (in $\mathbf{p}$) has been modified and does not make use of the iterate $\mathbf{r}_{k+1}$ computed (sequentially) "before" it, but instead uses the iterate $\mathbf{r}_k$. In this sense, this version may be thought of as a "Jacobi" version of the standard "Gauss–Seidel-like" conjugate gradient algorithm. The analysis of the standard conjugate gradient algorithm made above may be repeated almost verbatim, leading to the conclusion that the choices

$$\alpha_k = \frac{\langle \mathbf{r}_k, \mathbf{p}_k \rangle}{\langle \mathbf{A}\mathbf{p}_k, \mathbf{p}_k \rangle}, \quad \beta_k = -\frac{\langle \mathbf{p}_k, \mathbf{A}\mathbf{r}_k \rangle}{\langle \mathbf{p}_k, \mathbf{A}\mathbf{p}_k \rangle} \tag{2.158}$$

(the only difference is in the numerator of β_k) ensure that $\mathbf{r}_k$ is a decreasing sequence in $\mathbf{A}$-norm, and furthermore that $\|\mathbf{p}_{k+1}\|_{\mathbf{A}}^2 < \|\mathbf{r}_k\|_{\mathbf{A}}^2$, implying that $\mathbf{p}_k$ is also a decreasing sequence, although it decreases more slowly than it would in the standard conjugate gradient method (for which the inequality $\|\mathbf{p}_{k+1}\|_{\mathbf{A}}^2 < \|\mathbf{r}_{k+1}\|_{\mathbf{A}}^2$ was obtained). This confirms the conventional wisdom that "Gauss–Seidelization" is conducive to faster convergence, and indeed, numerical experiments confirm this.

The conjugate gradient algorithm interpreted as a dynamic controller

A block diagram representation is helpful in order to interpret what has just been done, in terms of the taxonomy of iterative methods proposed as well as making the controller structure explicit. Comparing the block diagrams of Figures 2.12 and 2.13, it becomes clear that, although the box representing the plant (i.e., problem or equation to be solved) has remained the same, the box representing the controller (i.e., solution method) is considerably more sophisticated with respect to the simple controllers studied in section 2.3. It is, in fact, a dynamic time-varying or nonstationary controller, in the spirit of the dynamic controllers considered for nonlinear equations in section 2.1. The upshot of the increased sophistication is that the method (conjugate gradient) is more efficient. In fact, it is well known that, in infinite precision, conjugate gradient is actually a direct method (i.e., it converges in n steps for an $n \times n$ matrix $\mathbf{A}$) [Kel95]. In control terms, this last observation can be rephrased by saying that the "conjugate gradient controller" achieves so-called dead beat control in n steps.

The backpropagation with momentum algorithm is the conjugate gradient algorithm

The well-known backpropagation with momentum (BPM) method (resp., algorithm) is a variant of the gradient or steepest descent method that is popular in the neural network literature [PS94, YC97]. In light of the control formulation of the conjugate gradient method presented in section 2.3.2, the BPM method for quadratic functions is now shown to be exactly equivalent to the conjugate gradient method, allowing derivation of a so-called optimally tuned learning rate and momentum parameters for the former method, for the nonstationary or time-varying case (as opposed to most of the literature in the field of neural networks, which treats only the time-invariant case).

Consider the problem of determining a vector $\mathbf{x}$ (thought of as a "set of network weights" in the neural network context) that minimizes a quadratic error function

$$f(\mathbf{x}) = \frac{1}{2}\langle \mathbf{x}, \mathbf{A}\mathbf{x}\rangle - \langle \mathbf{b}, \mathbf{x}\rangle + c, \tag{2.159}$$

where $\mathbf{A}$ is a symmetric positive definite matrix. The gradient $\nabla f(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} =: -\mathbf{r}$ is also called the residue (in the context of solution of the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$). In this notation, the BPM algorithm can be written as

$$\mathbf{x}_{k+1} - \mathbf{x}_k = \mu_k(\mathbf{x}_k - \mathbf{x}_{k-1}) + (1 - \mu_k)\lambda_k \mathbf{r}_k. \tag{2.160}$$

Clearly, the idea behind the BPM algorithm is to define the new increment $(\mathbf{x}_{k+1} - \mathbf{x}_k)$ as a (time-varying) convex combination of the old increment $(\mathbf{x}_k - \mathbf{x}_{k-1})$ and a multiple (λ_k) of the residue $(\mathbf{r}_k)$. The parameter λ_k is called the *learning rate*, while the parameter μ_k is called the *momentum factor* [TH02].

Notice that (2.160) is the same as (2.130), so that the BPM method for quadratic functions is exactly equivalent to the conjugate gradient method, for an adequate choice of the parameters μ_k and λ_k.

Indeed, the optimal CLF/LOC choice of the controls α_k, β_k immediately yields the optimal learning rate λ_k and momentum factors μ_k:

$$\begin{aligned} \alpha_k &= (1 - \mu_k)\lambda_k, \\ \alpha_k \gamma_k &= \mu_k. \end{aligned} \tag{2.161}$$

One can solve for the optimal learning rate and momentum factor in terms of the optimal choices of CG parameters α_k (2.151) and β_k (2.152):

$$\begin{aligned} \mu_k &= (\alpha_k/\alpha_{k-1})\beta_{k-1}, \\ \lambda_k &= \alpha_k\alpha_{k-1}/(\alpha_{k-1} - \alpha_k\beta_{k-1}). \end{aligned} \tag{2.162}$$

Equations (2.162) and (2.151) show that the optimal learning rate and momentum factor can be calculated in terms of the state variables $(\mathbf{r}, \mathbf{p})$, although this involves calculation of more inner products than the conjugate gradient method.

The overall conclusion is that it is easier just to use the standard conjugate gradient method, which has tried and tested variants, both linear and nonlinear [NW99], rather than use the equivalent "optimally tuned" BPM algorithm. Of course, there are many practical issues involved in determining ultimate performance, and the complexity and cost of each iteration of an optimal algorithm may offset its faster rate of convergence.

In section 2.3.3, a continuous version of the BPM or conjugate gradient method is proposed and analyzed and shown to be a version of the so-called heavy ball with friction method for continuous optimization.

Taxonomy of linear iterative methods from a control perspective

The block diagram representation has the virtue of allowing us to make a clear separation between the problem and the algorithm, making it easy to classify as well as generalize the strategies used in the algorithms. Taking the example of linear iterative methods, we see a progression of successively more complex controllers: constant (α), nonstationary or time varying $(\alpha_k\mathbf{I})$, multivariable $(\mathbf{K})$, multivariable time varying $(\alpha_k\mathbf{K}_k)$, and dynamic, leading to most of the standard iterative methods in a natural manner. For linear iterations, the results of this section lead to a "dictionary" relating controller choice to numerical algorithm that we present in the form of Table 2.5, which makes reference to Figure 2.1 and uses the terminology of [Kel95, SvdV00]. The standardized CLF analysis technique leads to the conventional choices of control parameters. It is worth noting that 2-norms, possibly weighted with a diagonal or positive definite matrix, usually work as CLFs. This is in sharp contrast with the situation for an arbitrary nonlinear dynamical system, for which, as a rule, considerable ingenuity is required to find a suitable CLF. Another consequence of the relative ease in finding quadratic CLFs is that each of these leads to a different algorithm, so that there is scope for devising new algorithms, showing that the CLF approach has an inherent richness. Moreover, the control Liapunov approach is easily generalizable to a Hilbert space setting, following the work on iterative methods for operators by Kantorovich [KA82], Krasnosel'skii [KLS89], and others.

Some disclaimers should also be made here. Although the control approach provides guidelines for algorithm design, it does not free the designer of the need for a careful analysis of issues such as roundoff error (robustness), computational complexity, order of convergence, etc. It should also be noted that many standard solutions of control problems are infeasible in numerical analysis because they would involve more computation for their implementation than the original problem. Here the challenge is for control theorists to develop limited complexity controllers, which, to some extent, driven by technological needs such as miniaturization and low energy consumption, are now being researched in control theory.

Table 2.5. *Taxonomy of linear iterative methods from a control perspective, with reference to Figure* 2.1. *Note that* $\mathcal{P} = \{\mathbf{I}, \mathbf{I}, \mathbf{A}, \mathbf{0}\}$ *in all cases.*

Controller $\mathcal{C}$	Controller type	Class of method	Specific methods
$\{\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{I}\}$	Static, stationary	Richardson	
$\{\mathbf{0}, \mathbf{0}, \mathbf{0}, \alpha_k\mathbf{I}\}$	Static, nonstationary	Adaptive Richardson	Chebyshev
$\{\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{K}\}$	Static, stationary	Preconditioned Richardson	Jacobi, Gauss–Seidel, SOR, extrap. Jacobi
$\{\mathbf{0}, \mathbf{0}, \mathbf{0}, \alpha_k\mathbf{K}_k\}$	Static, nonstationary	Adaptive preconditioned Richardson	
$\{\mathbf{0}, \mathbf{0}, \mathbf{0}, \alpha_k(\mathbf{r}_k)\mathbf{I}\}$	Static, nonstationary	Steepest descent	Orthomin(1)
$\{\beta_k\mathbf{I} - \alpha_k\mathbf{A}, \mathbf{I}, \mathbf{I}, \mathbf{0}\}$, in variables $\mathbf{p}_k, \mathbf{x}_k$	Dynamic, nonstationary	Conjugate gradient	Conjugate gradient, Orthomin(2), Orthodir
$\alpha_k, \beta_k,$ in variables $\mathbf{r}_k, \mathbf{x}_k$	Proportional-derivative, nonstationary	Conjugate gradient	Conjugate gradient, Orthomin, Orthodir
$\{\alpha_k\mathbf{I}, \mathbf{I}, \beta_k\mathbf{I}, \mathbf{0}\}$	Dynamic, nonstationary	Second-order Richardson	Frankel

2.3.3 Continuous algorithms for finding optima and the continuous conjugate gradient algorithm

The problem of unconstrained minimization of a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ can be viewed as the zero finding problem of the gradient of f, denoted $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$. From this perspective, the zero finding methods studied in this chapter can be used for unconstrained optimization. In this section, continuous methods based on dynamical systems will be studied. In particular, a continuous-time analog of the conjugate gradient method will be developed and studied using a CLF approach, and will be related to existing dynamical system methods that are referred to collectively as *continuous optimization methods*.

There are many different ways to write a continuous version of the discrete conjugate gradient iteration. One natural approach is to write continuous versions of (2.137) and (2.138) as follows:

$$\dot{\mathbf{r}} = -\alpha\mathbf{A}\mathbf{p}, \tag{2.163}$$

$$\dot{\mathbf{p}} = \mathbf{r} - \beta\mathbf{p}. \tag{2.164}$$

Elimination of the vector $\mathbf{p}$ yields the conjugate gradient ODE:

$$\ddot{\mathbf{r}} + \beta\dot{\mathbf{r}} + \alpha\mathbf{A}\mathbf{r} = \mathbf{0}. \tag{2.165}$$

Introducing the quadratic potential function

$$\Phi = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}, \tag{2.166}$$

for which

$$\nabla\Phi(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} =: -\mathbf{r},$$

allows (2.165) to be rewritten as

$$\ddot{\mathbf{r}} + \beta\dot{\mathbf{r}} - \alpha\mathbf{A}\nabla\Phi(\mathbf{x}) = \mathbf{0}. \tag{2.167}$$

Since $\mathbf{A} = \nabla^2\Phi(\mathbf{x})$, (2.167) can be written as

$$\ddot{\mathbf{r}} + \beta\dot{\mathbf{r}} - \alpha[\nabla^2\Phi(\mathbf{x})]\nabla\Phi(\mathbf{x}) = \mathbf{0}. \tag{2.168}$$

Observing that $\dot{\mathbf{r}} = -\mathbf{A}\dot{\mathbf{x}}$, $\ddot{\mathbf{r}} = -\mathbf{A}\ddot{\mathbf{x}}$, in $\mathbf{x}$-coordinates, (2.165) becomes

$$\ddot{\mathbf{x}} + \beta\dot{\mathbf{x}} + \alpha\nabla\Phi(\mathbf{x}) = \mathbf{0}. \tag{2.169}$$

In [Pol64], the idea of using a dynamical system that represents a heavy ball with friction (HBF) moving under Newtonian dynamics in a conservative force field is investigated. Specifically, the *HBF ODE* is

$$\ddot{\mathbf{x}}(t) + \theta\dot{\mathbf{x}}(t) + \nabla\Phi(\mathbf{x}(t)) = \mathbf{0}. \tag{2.170}$$

Clearly, (2.169) is an instance of the HBF method, where the parameters β (friction coefficient) and α (related to the spring constant) need to be chosen in order to make the trajectories of (2.165) tend to zero asymptotically.

The *steepest descent ODE* is defined as follows:

$$\dot{\mathbf{x}}(t) = -\epsilon\nabla\Phi(\mathbf{x}(t)). \tag{2.171}$$

As pointed out in [AGR00], the damping term $\theta\dot{\mathbf{x}}(t)$ confers optimizing properties on (2.170), but it is isotropic and ignores the geometry of Φ. Another connection pointed out in [AGR00] is that the second derivative term $\ddot{\mathbf{x}}(t)$, which induces inertial effects, is a singular perturbation or regularization of the classical *Newton ODE*, which may be written as follows:

$$\nabla^2\Phi(\mathbf{x}(t))\dot{\mathbf{x}}(t) + \nabla\Phi(\mathbf{x}(t)) = \mathbf{0}. \tag{2.172}$$

In the neural network context, $\mathbf{x}$ is the weight vector, usually denoted $\mathbf{w}$, and the potential energy function Φ is the error function, usually denoted $E(\mathbf{w})$ (as in [Qia99]). In fact, with these changes of notation, it is clear that the HBF equation (2.170) is exactly the equation that has been proposed [Qia99] as the continuous analog of BPM, using a similar physical model (point mass moving in a viscous medium with friction under the influence of a conservative force field and with Newtonian dynamics). Thus, the continuous version of BPM is the HBF ODE and may be regarded either as a regularization of the steepest descent ODE or as the classical Newton ODE.

Allowing variable coefficients into (2.165) gives

$$\ddot{\mathbf{x}} + \beta(\mathbf{x})\dot{\mathbf{x}} + \alpha(\mathbf{x})\nabla\Phi(\mathbf{x}) = \mathbf{0}, \tag{2.173}$$

where $\alpha(\cdot)$ and $\beta(\cdot)$ are nonnegative parameters to be chosen adequately in order that the trajectories of (2.173) converge to an equilibrium (i.e., a minimum of the potential or energy function).

In order to view this problem as a control problem amenable to treatment using a CLF, observe that (2.173) can be written as a first-order differential equation in the standard fashion, by introducing the state vector $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2) := (\mathbf{x}, \dot{\mathbf{x}})$ as

$$\dot{\mathbf{z}} = \begin{bmatrix} \mathbf{z}_2 \\ -\alpha(\mathbf{z})\nabla\Phi(\mathbf{z}_1) - \beta(\mathbf{z})\mathbf{z}_2 \end{bmatrix} =: \mathbf{f}(\mathbf{z}, \mathbf{u}(\mathbf{z})), \tag{2.174}$$

where $\mathbf{u}(\mathbf{z}) := (\alpha(\mathbf{z}), \beta(\mathbf{z}))$ is the control input to be designed, using a CLF, such that the zero solution of (2.174) is asymptotically stable.

Choice of continuous conjugate gradient algorithm parameters using LOC/CLF

From the discussion in section 2.3.2, it is natural to consider the discrete conjugate gradient iteration (2.137), (2.138) as a pair of coupled bilinear systems that are the starting point in the derivation of the parameters α and β, regarded as control inputs. We will repeat this approach in the analysis of (2.163) and (2.164), rather than simply analyzing stability properties of the second-order vector ODE (2.165) with variable parameters α and β. A control Liapunov argument similar to that in section 2.3.2 is used. The continuous-time analog of the conjugate gradient algorithm (section 2.3.2) is described in the following theorem.

Theorem 2.11. *Given the symmetric, positive definite matrix* $\mathbf{A}$ *and a quadratic function* $\Phi = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}$*, the trajectories of the conjugate gradient dynamical system, dependent on the positive parameters* α *and* β*, defined as*

$$\begin{aligned} \dot{\mathbf{r}} &= \alpha\mathbf{A}\mathbf{p}, \\ \dot{\mathbf{p}} &= \mathbf{r} - \beta\mathbf{p}, \end{aligned}$$

converge globally to the minimum of the quadratic function Φ *(i.e., to the solution of the linear system* $\mathbf{A}\mathbf{x} = \mathbf{b}$*) if the parameters* α *and* β *are chosen as follows:*

$$\begin{aligned} &\text{if } \langle \mathbf{r}, \mathbf{p}\rangle \neq 0, \quad \alpha = \langle \mathbf{r}, \mathbf{A}\mathbf{p}\rangle / \langle \mathbf{r}, \mathbf{p}\rangle, \quad \beta > 0, \\ &\text{if } \langle \mathbf{r}, \mathbf{p}\rangle = 0, \quad \beta \text{ such that } \langle \mathbf{r}, \mathbf{A}\mathbf{p}\rangle - \beta\langle \mathbf{p}, \mathbf{A}\mathbf{p}\rangle < 0, \end{aligned}$$

where $\mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}$*. The parameter* β *is chosen as follows: If the inner product* $\langle \mathbf{r}, \mathbf{A}\mathbf{p}\rangle$ *is positive, then* $\beta > \langle \mathbf{r}, \mathbf{A}\mathbf{p}\rangle / \langle \mathbf{p}, \mathbf{A}\mathbf{p}\rangle$*; if it is negative or zero, then any positive choice of* β *will do.*

Proof. Consider the CLF candidate

$$V(\mathbf{r}, \mathbf{p}) = \frac{1}{2}\langle \mathbf{r}, \mathbf{A}^{-1}\mathbf{r}\rangle + \frac{1}{2}\langle \mathbf{p}, \mathbf{A}\mathbf{p}\rangle. \tag{2.175}$$

Then

$$\begin{aligned}\dot{V} &= \langle \dot{\mathbf{r}}, \mathbf{A}^{-1}\mathbf{r}\rangle + \langle \dot{\mathbf{p}}, \mathbf{A}\mathbf{p}\rangle \\ &= \langle -\alpha \mathbf{A}\mathbf{p}, \mathbf{A}^{-1}\mathbf{r}\rangle + \langle \mathbf{r} - \beta \mathbf{p}, \mathbf{A}\mathbf{p}\rangle \\ &= -\alpha \langle \mathbf{p}, \mathbf{r}\rangle + \langle \mathbf{r}, \mathbf{A}\mathbf{p}\rangle - \beta \langle \mathbf{p}, \mathbf{A}\mathbf{p}\rangle,\end{aligned}$$

whence it follows that appropriate choices of α and β that make $\dot{V}$ negative semidefinite are as follows:

$$\text{if } \langle \mathbf{r}, \mathbf{p}\rangle \neq 0, \quad \alpha = \frac{\langle \mathbf{r}, \mathbf{A}\mathbf{p}\rangle}{\langle \mathbf{r}, \mathbf{p}\rangle}, \quad \beta > 0, \tag{2.176}$$

$$\text{if } \langle \mathbf{r}, \mathbf{p}\rangle = 0, \quad \beta \text{ such that } \langle \mathbf{r}, \mathbf{A}\mathbf{p}\rangle - \beta \langle \mathbf{p}, \mathbf{A}\mathbf{p}\rangle < 0. \tag{2.177}$$

Since β and $\langle \mathbf{p}, \mathbf{A}\mathbf{p}\rangle$ are positive, it follows that the choice of β in (2.177) depends on the sign of $\langle \mathbf{r}, \mathbf{A}\mathbf{p}\rangle$: If this inner product is positive, then $\beta > \langle \mathbf{r}, \mathbf{A}\mathbf{p}\rangle / \langle \mathbf{p}, \mathbf{A}\mathbf{p}\rangle$; if it is negative or zero, any positive choice of β will do. Since $\dot{V}$ is only semidefinite and α and β are functions of the state variables $\mathbf{r}$, $\mathbf{p}$, LaSalle's theorem (Theorem 1.27) can be applied. It states that the trajectories of the conjugate gradient flow (2.163) and (2.164) will approach the maximal invariant set $\mathcal{M}$ in the set

$$\mathcal{G} := \{(\mathbf{r}, \mathbf{p}) : \dot{V} = 0\}. \tag{2.178}$$

Invariance of $\mathcal{M} \subset \mathcal{G}$ means that any trajectory of the controlled system starting in $\mathcal{M}$ remains in $\mathcal{M}$ for all t.

Note that $\dot{V} = 0$ can occur only if (2.176) occurs and that this implies $\dot{V} = -\beta \langle \mathbf{p}, \mathbf{A}\mathbf{p}\rangle$, so that $\mathcal{G}$ can be alternatively characterized as $\{\mathbf{p} = \mathbf{0}\}$. From (2.163), $\mathbf{p} = \mathbf{0} \Rightarrow \dot{\mathbf{r}} = \mathbf{0}$, which, in turn, implies that $\mathbf{r}$ is constant. From (2.164), $\mathbf{p} = \mathbf{0}$ implies that $\dot{\mathbf{p}} = \mathbf{r}$. Since $\mathbf{r} = c$ (constant), this means that c must be zero (otherwise $\mathbf{p} = \mathbf{0}$ could not occur). Global asymptotic stability of the origin now follows from LaSalle's theorem (Theorem 1.27). □

Note that the Liapunov function could be chosen as

$$V(\mathbf{r}, \mathbf{p}) = \frac{1}{2}\langle \mathbf{r}, \mathbf{A}^{-1}\mathbf{r}\rangle + \frac{1}{2}\langle \mathbf{p}, \mathbf{Q}\mathbf{p}\rangle,$$

where $\mathbf{Q}$ is any positive definite matrix. In particular, the choice $\mathbf{Q} = \mathbf{I}$ results in the simple choice of parameters $\alpha = 1$, $\beta > 0$, demonstrating that the continuous version of the conjugate gradient method can even utilize constant parameters, as opposed to the discrete conjugate gradient method, where the "parameters" α and β must be chosen as functions of the state vectors $\mathbf{r}$ and $\mathbf{p}$. Notice, however, that in the continuous-time case, they are chosen either as constants or in feedback form, rather than being regarded as some arbitrary functions of time that must be chosen to stabilize (2.163) and (2.164). This makes it possible to use LaSalle's Theorem (Theorem 1.27) to obtain a global asymptotic stability result. The choice of α and β given in Theorem 2.11 corresponds to the choices made in the discrete conjugate gradient iteration. Note that a choice of initial conditions consistent with the discrete conjugate gradient iteration is $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ and $\mathbf{p}_0 = \mathbf{r}_0$.

There is, of course, a close connection between these second-order dynamical systems for optimization and the second-order dynamical systems for zero finding proposed in (2.57). In fact, if, for example, the algorithm DC1 from Table 2.2 is applied to the problem of finding the zeros of the gradient ∇f (which is the minimization problem studied at the beginning of this section), then (2.57) for algorithm DC1 becomes

$$\ddot{\mathbf{x}} + \mathbf{\Gamma}\dot{\mathbf{x}} + [\nabla^2 f(\mathbf{x})]\nabla f(\mathbf{x}) = \mathbf{0}, \tag{2.179}$$

noting that the Hessian $\nabla^2 f(\mathbf{x})$ is a symmetric matrix. Comparing (2.168) and (2.179), it becomes clear that the algorithm DC1 is an instance of the HBF method; however, the former is derived from the CLF method, without recourse to mechanical analogies. Algorithms DC3 and DC4 are, to the best of our knowledge, new second-order dynamical systems for the zero finding/optimization task, and further research is needed to verify their effectiveness.

In closing, we call the reader's attention to two quotes. The first is a paragraph from Alber [Alb71] that is as relevant today (with some minor changes in the buzz words) as when it was written three decades ago:

> *The increasing interest in continuous-descent methods is due firstly to the fact that tools for the numerical solution of systems of ordinary differential equations are now well developed and can thus be used in conjunction with computers; secondly, continuous methods can be used on analog computers [neural networks]; thirdly, theorems concerning the convergence of these methods and theorems concerning the existence of solutions of equations and of minimum points of functionals are formulated under weaker assumptions than is the case for the analogous discrete processes.*

Similar justification for the consideration of continuous versions of well-known discrete-time algorithms can be found in [Chu88, Chu92].

The second quote is from Bertsekas's encyclopedic book [Ber99] on nonlinear programming:

> *Generally, there is a tendency to think that difficult problems should be addressed with sophisticated methods, such as Newton-like methods. This is often true, particularly for problems with nonsingular local minima that are poorly conditioned. However, it is important to realize that* often the reverse is true*, namely that for problems with "difficult" cost functions and singular local minima, it is best to use simple methods such as (perhaps diagonally scaled) steepest descent with simple stepsize rules such as a constant or diminishing stepsize. The reason is that methods that use sophisticated descent directions and stepsize rules often rely on assumptions that are likely to be violated in difficult problems. We also note that for difficult problems, it may be helpful to supplement the steepest descent method with features that allow it to deal better with multiple local minima and peculiarities of the cost function. An often useful modification is the heavy ball method. . . .*

2.4 Notes and References

Continuous-time dynamical systems for zero finding

Continuous algorithms have been investigated in the Russian literature [Gav58, AHU58, Ryb65b, Ryb69b, Ryb69a, Alb71, Tsy71, KR76, Tan80] as well as the Western literature [Pyn56, BD76, Sma76, HS79] and the references therein. More recently, Chu [Chu88] developed a systematic approach to the continuous realization of several iterative processes in numerical linear algebra. A control approach to iterative methods is mentioned in [KLS89], but not developed as in this book.

Other terms for continuous algorithms that have been, or are, in fashion, are analog circuits, analog computers and more recently, neural networks.

Tsypkin [Tsy71] was one of the first to formulate optimization algorithms as control problems and to raise some of the questions studied in this book. An early discussion of the continuous- and discrete-time Newton methods can be found in [Sil80]. The discrete formulation of CLFs is from [AMNC97], which, in turn, is based on [Son89, Son98]. The idea of introducing a taxonomy of iterative methods in section 2.3 is mentioned in [KLS89], although, once again, the discussion in this reference is not put in terms of CLFs, and there is only a brief mention of control aspects of the problem. Variable structure Newton methods were derived by the present authors in [BK04a].

The continuous Newton method and its variants have been the subject of much investigation [Tan80, Neu99, DK80a, DK80b, DS75, Die93, RZ99, ZG02, HN05]. Branin's method, a much studied variant of the continuous Newton method, was originally proposed in [Dav53a, Dav53b] and, since then, has been studied in many papers: [ZG02] contains many references to this literature. Path following, continuation, and homotopy methods and their interrelationships and relationship to Branin's method are discussed in [ZG02, Neu99, Qua03].

The so-called gradient enhanced Newton method is introduced in [Gra03], where its connections with the Levenberg–Marquardt method are discussed. The latter method is a prototypical team method, which combines two (or more) algorithms, in an attempt to get a hybrid algorithm that has good features of both component algorithms. In Barán, Kaskurewicz, and Bhaya [BKB96] hybrid methods (resp., team algorithms) are put in a control framework and their stability analyzed in the context of asynchronous implementations.

CLF technique

As far as using Liapunov methods in the analysis of iterative methods is concerned, contributions have been made in both Russian [EZ75, VR77] and Western literature [Hur67, Ort73]. The generalized distance functions used in [Pol71, Pol76] can be regarded as Liapunov functions, as has been pointed out in [Ber83]. The Liapunov technique is extremely powerful and can be used to determine basins of convergence [Hur67], as well as to analyze the effects of roundoff errors [Ort73] and delays [KBŠ90].

Early use of a quadratic CLF to study bilinear systems occurs in [Qui80, RB83]. Recent descriptions of the CLF approach and the LOC approach can be found in [Son98, VG97], respectively.

Iterative methods for linear and nonlinear systems of equations

The classic reference for iterative methods for nonlinear equations is [OR70]; a recent survey, including both local and global methods, is [Mar94].

The interested reader is invited to compare the control approach to the conjugate gradient algorithm developed above with other didactic approaches, such as those in [SW95, She94], or an analysis from a z-transform signal processing perspective [CW00]. In our view, the control approach is natural and this is borne out by its simplicity.

Continuous-time systems for optimization

The discussion of the continuous version of the conjugate gradient algorithm and its connection to the well-known BPM method (much used in neural network training) is based on [BK04b], which also contains an analysis of the time-invariant continuous conjugate gradient algorithm.

The BPM algorithm has been much analyzed in the neural network literature, both theoretically and experimentally (see, e.g., [YCC95, YC97, KP99, PS94, HS95] and references therein). The paper [BK04b] shows that the analyses of Torii and Hagan [TH02] and Qian [Qia99] of the time-invariant BPM method are special cases of the conjugate gradient method.

Early papers on "continuous iterative methods" and "analog computing" (see, e.g., [Pol64, Ryb69b, Tsy71]) proposed the use of dynamical systems to compute the solution of various optimization problems. A detailed analysis of the HBF system is carried out in [AGR00], and further developments are reported in [AABR02].

Relationship between continuous and discrete dynamical systems

This is a large subject: Aspects of asymptotic and qualitative behaviors of continuous dynamical systems and their discrete counterparts are treated in the monograph [SH98] and, from a control viewpoint, in [Grü02].

The relationship between convergence rates of discrete and continuous dynamical systems is treated in [HN04].

Another important aspect of the relationship between a continuous dynamical system and its associated discrete dynamical system has to do with the discretization or integration method used. Different integration methods for continuous algorithms are discussed in [Bog71, IPZ79, DS80, Bre01].