Decentralized Phase Regulation of Cyclic Robotic Systems

by

Eric Klavins

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2001

Doctoral Committee:
Professor Daniel E. Koditschek, Co-Chair
Professor William C. Rounds, Co-Chair
Professor Anthony Bloch
Professor Kevin J. Compton
Professor Pramod Khargonekar

To Laurie

ACKNOWLEDGMENTS

I thank my advisor Dan Koditschek for his enthusiasm and vision, for our inspiring conversations, for his help with many technical details and especially for his patience and honesty. Dan welcomed me into his amazing group of graduate students and researchers and created many invaluable opportunities for me to work with and learn from his collaborators at universities around the world. I thank my co-advisor Bill Rounds who got me started at the University of Michigan and introduced me to many beautiful ideas which helped shape my somewhat unconventional approach to robotics.

I thank my undergraduate professors Gerald Eisman, for encouraging me to go to graduate school, and Robert Douglas, for giving me the highest grade (a B+) in his Discrete Mathematics class.

I thank Al Rizzi and the rest of his group at the Microdynamic Systems Laboratory at CMU. Al's work on robotic juggling was the first truly amazing thing I saw at graduate school and the *Minifactory* that he and others are building continues to raise fundamental questions. I thank my collaborator Robert Ghrist, who is an excellent mathematician at the Georgia Institute of Technology, for answering my simple minded questions about topology. I thank Martin Buehler and his group at McGill University for their pragmatism.

I thank the other graduate students in Dan's group, especially Haldun Komsuoglu for the unending stream of ideas that pours forth from his brain and Uluc Saranli for his inspirational technical prowess and for building RHex the robot. Many ideas float around the lab, arising in ongoing conversations about the future of science and engineering, and it is impossible to keep track of their origins. I thank Rick Groff, Noah Cowan, Joel Weingarten and Bill Schwind for many such conversations.

I thank my good friend Chris Brooks for more reasons than I can list here but especially for our arguments about the meaning of intelligence (the jury is still out).

I thank the staff at Advanced Technology Laboratory here at the University of Michigan, especially Wendy Gillikin and Karen Alexa, without whom none of us would be able to get anything at all done.

Most importantly, I thank my family. My parents Andrew and Dorothy have without question supported and encouraged me throughout my life, no matter what strange thing I decided to do next. No one could ask for more. I thank my son Dylan whose delightful smile can instantly erase any setback. And finally, my eternal gratitude goes to my wife Laurie, who has never lost confidence in me. I am glad I am her partner in our many adventures together.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF APPENDICES

APPENDIX

CHAPTER 1

Introduction

The goal of this thesis is to begin to make formal an idea which is already used intuitively and with great success in the laboratory. The idea is to leverage the knowledge of how to control a machine to perform a set of basic tasks toward the design of a controller that makes the machine do more complex tasks. We say that the more complex task is *composed* of the basic tasks. For example, a machine that can be controlled to hop on one leg should be adaptable to one that hops by alternating between two legs — without the need to redesign the entire system from first principles. A general theory of controller composition would apply to many situations in engineering, greatly increasing in size the set of tools available for design synthesis and system verification — just as packaged microchips and modular computer code have greatly eased the task of designing electronic devices and software. It may also apply to systems outside of engineering such as biomechanics (as alluded to at the end of Chapter 4), which attempts to explain how animals control their movements. Controller composition is thus a fundamental problem, requiring tools from dynamical systems, control theory, robotics and computer science. It is also a difficult problem in general, because of the intricate ways in which controlled dynamical systems may couple when operating in combination. Therefore, in this thesis, we focus on a specific kind of subsystem and type of composition — the parallel composition of cyclic systems — and stay mostly within the realm of robotics, with only a few remarks on applications in other fields.

Many tasks in robotics require that cyclic actions be coordinated. A walking or running robot must coordinate its legs, which repeatedly convert motor torque into forward thrust, according to some desired gait — a schedule for the delivery of leg thrusts. Thus, the legs must deliver their thrusts according to some rhythm, whether simultaneously, in alternating groups of three, or some other pattern. If each leg is supposed to have some degree of independence — for example, to deal with terrain variations local to it — then, in the absence of some coordination mechanism, the phase relationships between the legs will surely destabilize. In this thesis we address the problem of modifying the controller of an individual cyclic subsystem (e.g., a leg, or motor subunits corresponding to some modular constituent of an ensemble) so that it uses information obtained from other subsystems (e.g. the other legs) in such a manner that the entire system (e.g., the robot) stabilizes around a desired phase relationship (e.g., a gait).

In more formal terms, the thesis addresses the problem of composing, "in parallel," limit cycles. Namely, we assume we are presented with a number of independent controlled systems that each, separately, exhibit a limit cycle — an attracting periodic orbit. We seek a procedure for coupling together their controllers in such a fashion that the resulting coupled closed loop system exhibits a single limit cycle corresponding to a specified relationship among the original decoupled systems. The procedure should ideally be formal — relying simply on the presence of the individual attractors, independent of the details of the mechanisms that stabilize the individual subsystems. It should also be correct — accompanied by an automatic formal proof that the coupled system does indeed exhibit the specified periodic attractor.

The goal of this thesis is to introduce tools that rely on feedback, introduce a minimum number of communications links among components and require no central control signal. The hypothesis is that these tools are best thought of compositionally [42, 39, 40] — that is, that components or groups of components that behave correctly in isolation may be composed by altering their control algorithms in a formulaic fashion to achieve coordination. The resulting modularity should give rise to scalability, enabling the design of large, complex nonlinear and robust controlled systems.

1.1 Motivation

Formal techniques for building decentralized control architectures with provable properties will become increasingly important as robots and, more generally, physically embedded computing systems, become more and more complex [78]. A simple, three degree of freedom juggling robot [73], for example, may contain several processors which acquire sensory data, compute estimates and deliver control commands. A complex automated factory may require the coordination of hundreds of robots each performing processing, sensing and actuating locally [71, 63]. If, as seems increasingly likely, MEMS devices for distributed manipulation [27] can be used for assembly then possibly thousands or millions of microscopic actuators will require coordination. Ideally, to design controllers for such systems, we desire a provably correct method for controlling each component, using local information as well as information from neighboring components, so that some global task is performed by the entire collection of components. Similar goals have been realized, to some extent, in distributed computing [56]. However, in physically embedded computing systems, any control methodology, distributed or otherwise, must account for real-world mechanical forces, inaccurate sensing and a partially actuated environment. Addressing these latter complications is the main challenge to decentralized, modular control.

We are convinced that compositional methods for physical systems are possible. Work by Koditschek and collaborators has led to several functioning (laboratory) robots that possess relatively sophisticated dexterous behaviors formed by coupling previously working simpler constituents in an analogous manner. For example, robot "batters" have been constructed in compositions involving a one degree of freedom "vertical" limit cycle controller inspired by Raibert's hoppers [46]. A planar version of the batter is composed of the Raibert vertical cycle[1] with a one degree of freedom [13] "horizontal" fixed point controller that might be loosely construed as a nonlinear PD (that is, a variant on the traditional proportional-derivative design [44]). A subsequent spatial version is composed of the same Raibert vertical cycle with a two degree of freedom horizontal generalized PD [73]. These behaviors inspire this work and their design is reviewed in Chapter 2, which also contains a more general discussion of composition.

Several desirable consequences quickly follow when a composition technique for embedded controllers is defined in a manner that is both formal and correct. As discussed in Chapter 2, the "sequential composition" of point attractors enjoys a formal and completely general control theoretic semantics via graphs that locate attractors relative to their neighbor's basins (as delimited, for example, by their Lyapunov functions) [15]. Although this notion is conceptually simple, it is nevertheless quite useful: A well established tradition of sequential composition in AI — pre-image backchaining [54] — can be rendered formally and correctly in this manner. Formal composition methods can also assure scalability. For example, the author of this thesis has developed methods for composing hybrid factory robot programs into concurrent, multi-robot systems [42], which are reviewed at the end of Chapter 2. Based on this method, a provably correct compiler [40] has been built f or (a simplified, simulated version of) a modular factory [71].

In contrast to the compositions just cited, the parallel composition of cycles, the focus of this thesis, remains a less developed area. We have some experience with successful empirical efforts of this kind. The batters, above, have been joined to produce "jugglers" — again, both planar [14] and spatial [73] versions — by properly "interleaving" the two constituent vertical cycles. However, no correctness proof has heretofore existed, much less a clear report of any formal coupling principles suitable for applications in other domains. It is toward such a formal coupling principle that this thesis is addressed.

1.2 Specific Contributions

Parallel compositions of the sort described above, involving point attractors or points and one cyclic attractor, satisfy the following steady state behavior: the attractor of the coupled system is the cross product of the component attractors. In contrast, the obvious first complication arising from parallel constructions of several

[1] In the case of the batter, the vertical limit cycle corresponds to an attracting orbit in the position/velocity plane of the ball. As discussed in Chapter 4, this cycle is regulated in a manner similar to the hopping cycle in Raibert's work.

cyclic attractors is that their cross product is not directly related to any of the components, defining, instead, an n-torus, which does not in itself correspond to any natural robotic task that we have encountered. A composition of limit cycles resulting in a limit cycle, therefore, must also specify the limiting *phase relationships* desired of the components. The central contribution of Chapter 3 is to propose a simple but general formalism for specifying these phase relationships along with a general method for realizing the specification with a reference dynamical system. A number of example problems are then given to suggest how these reference dynamics might eventually be the basis of a formal composition technique for a large class of physically relevant cyclic systems.

We observe in Chapter 3 that any cyclic component system can be described in terms of *phase coordinates* consisting of phase and phase velocity, thus giving each system the same "interface" and introducing the model space of the composition — the n-torus. We then propose a simple means of specifying the desired phase relationships with a *connection graph*, whose nodes denote the component cyclic systems and whose edges, labeled "in phase" or "out of phase," denote their desired phase differences at steady state. Next, we present a general procedure for building a gradient-like vector field on the phase space whose coupling functions are governed by the connection graph. Finally, we identify algebraically the entire forward limit set of the closed loop coupled system, affording, thereby, a check that the desired limit cycle is essentially globally asymptotically stable.

We use the reference dynamics to compose several one degree of freedom cyclic components in four progressively more complicated examples evocative of the physically functioning juggling and running robots introduced above. These examples are chosen to suggest the range of control affordance that a more general composition procedure must encompass. It is well established in control theory that differences in actuation structure present radically different opportunities for point stabilization, ranging from arbitrarily achievable dynamics (e.g., in the case of fully actuated first order systems) to systems that are smoothly unstabilizable (e.g., in the case of the nonholonomically constrained mechanisms [9]). The corresponding stabilizability classification for limit cycles is far less established, hence we select examples that correspond to physical settings of interest. The examples illustrate how the same abstract connection graph can serve as the basis for compositions of cyclic systems of various types, each type corresponding to an intuitively different affordance over the subsystems' phases.

The first example is in fact essentially the same as the reference dynamical systems defined in Chapter 3. We consider highly abstracted components, each taking the form of a first order fully actuated subsystem — in phase coordinates, $\dot{\theta}_i = f_i(\theta_i, v_i), i \in \{1, ..., n\}$, where f_i is invertible with respect to the control, v_i, for each *phase*, $\theta_i \in S^1$. Since appropriate "inverse dynamics" style state feedback, $v_i = f_i^{-1}(\theta_i, u_i)$, can arbitrarily reshape the vector field, $\dot{\theta}_i = u_i$, it is clear that the phase θ of each component of the system is directly controllable. Such models are not uncommon in the coupled oscillator literature, and, for purposes of this paper, offer a simple setting in which to illustrate the application of the proposed composition technique. Specifically, in Section 3.4 we construct coupling controllers arising from two different connection graphs (Figure 3.1) for a set of six first order cyclic components. We use the algebraic characterization of the forward limit set on the cross product space,

$$\mathbb{T}^6 = \underbrace{S^1 \times S^1 \times ... \times S^1}_{6\ times}$$

to discover that the first connection graph is effective while the second is not.

In general, physical systems will not offer such direct affordance over phase velocity. For example, in action-angle coordinates [4], (E_i, θ_i) — the classical representation of phase for the one degree of freedom mechanical systems of present interest — a lossless closed loop component exhibits the dynamics $\dot{\theta}_i = \omega_i(E_i)$ and $\dot{E}_i = 0$. Physically realizable control inputs act on the acceleration variable of such a system and typically enter both the action and the angle dynamics in a complicated manner,

$$\begin{aligned} \dot{\theta}_i &= \omega_i(E_i, u_i) \\ \dot{E}_i &= f_i(\theta_i, E_i, u_i). \end{aligned}$$

However, many cyclic component systems of interest in robotics — in particular, all of those mentioned in the motivation section above — are regulated by means of periodically applied changes in "stiffness" or some similar physical variable that adjusts the action on a cycle-by-cycle basis because they can be actuated only intermittently. The term *intermittent* means that during certain periods, there is no affordance at all, that is, $f_i \equiv 0$, and $\partial\omega_i/\partial u_i \equiv 0$ except when θ_i is contained in some *contact set* $\mathcal{A}$. When such a subsystem operates in isolation, it is sufficient to verify that the "stiffening schedule" stabilizes a specified periodic orbit with respect to any convenient coordinate system. However when conceived as a component, the subsystem's stiffening schedule on the contact set must be altered with the dual purpose of regulating its own limit cycle as well as its phase relative to its specified neighbors.

In Chapter 4 we introduce two different examples of this situation, evocative, respectively, of the batting [14] and hopping robots [46] discussed above. In both cases we begin by introducing variants of the component subsystem controllers that were deployed on the physical machines: They yield asymptotically stable limit cycles for the component one degree of freedom systems with user-specified steady state total energy. We next address the parallel composition of two such cyclic components, defining the identical reference dynamics (on the two dimensional torus) as a model for juggling in the first example, and for bipedal hopping in the second. For both examples, we show how to compute the phase coordinate transformations required to implement the composition. The reference dynamics then serves to guide modifications in the subsystem controllers that confer sensitivity to relative phase in addition to the prior affordance over individual energy (now interpreted as phase velocity, through the action variable, in the new coordinate systems). We finally show, in both cases, that the resulting coupled systems do indeed achieve the desired closed loop behaviors.

Similar though they are, these two settings introduce a further distinction in subsystem actuation structure: the recourse to deadbeat as opposed to asymptotically stabilizing control. For the juggling example, because the contact set is a lower dimensional surface visited instantaneously each cycle, it is easy to build a subsystem controller that works in a "deadbeat" manner, achieving the desired ball height for an isolated single ball after one hit. In contrast, for the hopping example, the component controllers for the two isolated subsystems achieve asymptotic stability with convergence in infinite time. Despite this difference in input structure, the composition procedures are quite similar. A comparison of Equations 4.19 and 4.21, which define the return maps of these systems near equilibrium, shows that the second system incorporates a delay term, g, which marks the difference (and complicates the analysis). That our method applies equally well to each example is a hint of its possible generality.

A second key difference between these two examples underscores an additional feature of the composition method whose proper consideration lies beyond the scope of the present thesis. The gradient-like reference dynamics on the two dimensional torus entails two invariant cycles — the desired (essentially global) 180^o-out-of phase attractor and an in-phase repeller. The latter structure can be considered a dynamical "obstacle" to be avoided by the coupled system. In synchronized hopping, there is a dedicated actuator for each cyclic system (i.e., each hopper can control itself), while in juggling, a single actuator must switch its attention from each ball in the system, necessitating a sort of interleaving control. Separating the phase of the balls thus becomes an important part of the composition and is naturally encoded by the presence of the repeller.[2]

A final example of how reference dynamics can be used to couple cyclic systems of various sorts is presented in Chapter 5, where we build a decentralized controller for a hexapod robot called RHex. In this example, an internal, "ideal" six-oscillator network of the kind described in Chapter 3 is coupled with the six leg controllers of the robot. Each leg controller attempts to achieve the phase velocity of a corresponding internal oscillator while also feeding its own phase back to the internal oscillator. The resulting overall control scheme can be tuned with respect to the amount (or strength) of coupling between the environment and the internal oscillator network.

A formal treatment of a final point of similarity in all of these examples also lies beyond the scope of this thesis. The systems we design using the connection graph (Section 3.1) are decentralized, or modular, in form. The connection graph, in addition to specifying the desired phase relationships between systems, effectively specifies the desired communications structure to be used as well. Although we have

[2]We accomplish the change of attention of the paddle among the balls using an *attention function* (see Equation 4.15), the details of which are contained in Appendix A.

not developed a mature notion of decentralization (one that accounts for information flow or communication costs, for example), the present work provides basic tools for conceptualizing and synthesizing cyclic systems with various decentralized structures. These tools are illustrated by the application of the criterion supplied by Corollary 3.3.1 to Examples 3.4.1 and 3.4.2.

In summary, this thesis addresses a small step toward the general goal set forth above. We introduce a general formalism for specifying the steady state cyclic behavior of a collection of cyclic components. Treating the component subsystems as oscillators suggests a way to compose them using feedback terms based on the phases of neighboring oscillators to achieve the specified coupled behavior. If the components of the system are continuously actuated (i.e., they have direct control over their phase), the method is straightforward. If, on the other hand, the components have intermittent control over their phases (as with a paddle bouncing a ball or a leg delivering thrust to a robot), it is less obvious that our method applies. Nevertheless, we demonstrate that out method is applicable to this important class of problems. The similarity of method in the different examples strongly suggests that a general formalism of the kind we desire should be possible to define and practice.

1.3 Related Work

In this section we review related work on decentralized control and coupled oscillators. A more detailed review of the compositional approach is presented in Chapter 2.

Decentralized Control: As we have suggested, by decentralized control of a system $\dot{\mathbf{x}} = f(\mathbf{x}, u)$ we mean first that $\mathbf{x}$ can be broken into a number of subsystems $\mathbf{x} = \mathbf{x_1}; ...; \mathbf{x_n}$ where the semicolon means vector concatenation. Second, we require that controllers can be found so that for each subsystem i we have that

$$\dot{\mathbf{x}}_{\mathbf{i}} = f_i(\mathbf{x}_{j_1}, ..., \mathbf{x}_{j_k}, u_i(\mathbf{x}_{j_1}, ..., \mathbf{x}_{j_k})), \tag{1.1}$$

where the set $\{j_1, ..., j_k\}$ of neighbors of i is a proper subset of $\{1, ..., n\}$ (i.e., the control law for the ith system depends only on the neighbors of i). A basic example of such a system is the robot flocking behavior [68] in which multiple robots in a shared environment are each given the task of, for example, moving to a goal location while avoiding other robots. In many algorithms, a robot pays attention only to its closest neighbors, which may change; thus, this situation fits loosely into our framework. Issues of deadlock in these systems are the most difficult to resolve. To date, such methods are used without formal proofs of correctness, although in extremely simplified settings we have been able to show that certain strategies are correct [42, 39].

More obviously relevant to the present study are various decentralized control schemes for legged locomotion [6, 16] that are inspired by biological models of the stick insect [25] and other animals [26]. In such schemes, each leg of a six-legged robot is considered to be animated by a separate processor and actuator, deciding what to do based on the state of certain neighboring legs. Of course, the state of the robot's mechanical body depends upon the positions and velocities of its center of mass frame as well as the states of all the legs; thus, the system is decentralized only in the sense that the control of actuated components is decentralized. In [43] we presented a simulation study suggesting that the methods in this thesis can exhibit behaviors similar to those observed in a simple model [16] of this kind of coordination. We believe, but have not yet explicitly demonstrated, that the framework introduced here promotes a parsimonious view of the stick insect models, and one for which correctness of coordination proofs will be tractable. In this thesis it is proved that similar coordination strategies are robust and stable, albeit in a vastly simpler setting. We believe such a parsimonious and sound foundation is a necessary precursor to the widespread adoption of any walking or coordination strategies in robot design.

Our general approach to building decentralized systems, inspired by the traditions of computer science [1, 20], is compositional [15, 42, 39]. That is, we seek methods for composing previously isolated subsystems into more complex systems via adjustments of their individual controllers as we have already illustrated in our earlier remarks about "sequential" [15], "parallel" [13] and "interleaved" [73] control. There are many other informal uses of composition in robotics as well, such as the bottom up generation of flocking behaviors [68], for which formal compositional treatments could be quite useful as well. Chapter 2 is devoted to the exploration and review of the compositional approach in more detail.

Our simple characterization of a "decentralized" system suffers from the same sort of ambiguity that afflicts some computer science characterizations of distributed systems. That is, in their simplest forms the available formalisms (differential equations and computational complexity) are blind to the issues of information flow and communication costs just as are simple models of distributed computation (of course, many of these problems in computer science have been addressed [56]). We do not address these problems here, although we believe they are of tremendous importance as systems become more complex. In [43] we begin to address the tradeoff between centralized control structures, such as that used in our hexapod robot [74] and decentralized control.

Coupled Oscillators: The study of the nervous system has inspired a significant volume of work on coupled oscillators. The goal of such work has been to devise analytically tractable models of the neuron and of collections of neurons for the purpose of explaining observed neurophysiological phenomena. The typical approach is to treat neurons as oscillators of some kind and introduce various coupling terms [31]. Researchers have examined various regular topologies of such couplings as well (as for example in [47, 28]).

Oscillators appear in robotics whenever coordinated cyclic movements are required. As robotics researchers increasingly turn to biology for inspiration, coupled oscillators are used synthetically, to engineer control algorithms, rather than as modeling tools. Generally one finds two apparently complementary approaches: feedforward shape generators and feedback-driven networks of coupled oscillators. The first approach is inspired by the discovery of biological *central pattern generators* — oscillating groups of neurons that have been shown to produce rhythmic movements originating in the central nervous system [64, 22, 21] that seem associated with a feedforward style of control. In robotics, pattern generators are generally used to control repetitive movements. In the RHex hexapod robot discussed in Chapter 5, a simple first order linear oscillator is used as a "clock" to generate an alternating tripod gait. Similar mechanisms are used to control the shape of snake-like robots [61, 8] and in an underactuated two degree of freedom suspended leg [52] to produce feedforward locomotion. In contrast, feedback methods, such as those inspired by the previously mentioned study of the slow-moving stick insect [25], which have begun to be analyzed [16], couple internal oscillators with mechanical oscillators such as legs. The internal oscillators are then synchronized via some coupling function, yet are constantly disturbed by signals from the legs. In [36] a nonlinear oscillator model with feedback is used to coordinate leg motions in a biped.

In this thesis, synthesis of coordination control algorithms (rather than, say, the modeling and analysis of complex biological systems) remains the primary goal, hence we take the view that oscillators should be abstracted to phase and phase velocity coordinates. Thus, instead of using a complex, nonlinear oscillator to generate motions, we explicitly use simple gradient-like reference fields and concentrate on the means by which variously composed controllers may be introduced into variously configured control systems to produce coordinated movement in mechanical systems. A similarly synthetic view is used in [65] to synchronize clocks in a mesh-connected network of processors, but is otherwise rare.

1.4 Overview

In Chapter 2 we review the compositional approach to robotics and computer science. A loose distinction between *general* and *careful* composition is used to categorize various approaches with respect to the breadth of the class of objects to which they are applicable, and the point is made that careful compositions on only those systems satisfying some limiting property are usually more tractable. We also review in more detail the work on juggling and hopping by Koditschek and his former students [15, 14, 12, 73, 46], which was the main impetus for the present work. At the end of Chapter 2 we review in some detail active research on manufacturing and distributed assembly by the author of this thesis — research that is highly compositional and related in inspiration, if not in any as yet recognizable formal detail, to the present work on phase regulation.

In Chapter 3 we introduce a means of designing the *reference fields* first discussed in Section 1.2. We define a simple two dimensional reference field and then define a certain class of n dimensional reference fields based on connection graphs. The chapter ends with several examples of such reference fields and analyses of their behaviors.

In Chapter 4 we apply reference fields to the intermittent control tasks of ball batting and hopping. We first explain how to deal with one ball batted by a piston and one hopper. Then we combine two balls and then two hoppers. Finally we analyze the stability of the composed systems and examine some numerical simulations.

In Chapter 5 we describe a decentralized, phase regulation controller for the six legged, cockroach-like robot RHex and give the results of experiments in which we explore the controller's efficiency.

At the end of each of Chapters 3, 4 and 5 is a discussion of each chapter's content with an emphasis on things left undone. The final chapter, Chapter 6, is a higher level reflection on what was attempted and what was in fact accomplished with this thesis, as well as a review of some outstanding questions raised.

The appendices contain details not included in the main body of Chapter 4 on intermittent contract systems: the important construction of attention functions for the juggler, the tedious but routine derivation of the return map and period for our model of a hopping robot and the details of the proof of stability of the synchronized hopping system.

CHAPTER 2

The Compositional Approach: A Review

The idea behind the compositional approach is simple. A designer should not have to start from scratch when building complicated systems. Instead she should leverage the knowledge of how to make a machine do A and also how to make it do B in order to make a machine do A *and* B in some combination. For example, a program that controls a robot to drill a hole in a piece of sheet metal should be easily modified into a program that controls a robot to drill two holes in sequence or a program that controls two robots to drills holes simultaneously. A hopping controller for a one legged robot should be extensible to a hopping controller for a two legged robot that alternates between the legs. The difficulty with composition arises when the components to be composed interact in complex ways. This interaction is simple in the example of sequential drilling: the second instance of the drilling program must wait for the first instance to finish. On the other hand, a single legged hopping controller may destabilize in the context of two legged hopping due to the presence of the other leg — unless the controller is modified in some manner. In general, a component must be able to conform to a *compositional rule* that specifies how it should compensate for the behavior of any other component with which it may be composed.

The compositional approach to robotics, and embedded systems more generally, is inspired by the success of composition in computer science, particularly with respect to sequential programming, models of concurrency and the specification of systems. A sequential programming language, given by a grammar, is by nature compositional. For example, the statement, "`if` P `then` S_1 `else` S_2" is *composed of* the statements S_1 and S_2 and the predicate P using the "if-then" composition. Roughly, the components in a programming language are statements, and the compositional operators are the standard sequential, branching and looping constructs. In allowing the programmer to build very complex programs from simple parts lies the power of programming languages. Composition is used in concurrent systems as well where the basic components are processes. Systems for modeling concurrent systems are based on building simple processes and then combining them in various ways [5, 35, 57, 69]. In these approaches, a simple process A is given by a sequence of states (or actions) and processes are combined, among other ways, in sequence $A; B$ or in parallel $A_1 || A_2$ ("in parallel," of course, can mean many things and so must be precisely defined). A more general approach to composition is to compose *specifications* of systems [1, 19], where the more general idea of a "system" refers to a possibly electromechanical artifact as well as the program that controls it. A specification is essentially a description of how a system should behave, possibly as a function of its environment. When two systems are composed, however, each becomes a part of the other's environment. The challenge then becomes how to program components in isolation while reasoning about their behavior in concert.

Apparently, then, the compositional approach to designing systems consists of two parts. First is the design of a set of basic system components and specifications of their behaviors. Second is a theory of how the basic components may be combined, in various useful ways, which includes specifications of how the possible combinations behave. The kind of statements one would like to prove are of the following form: Anytime property P_1 holds for system S_1 and property P_2 holds for system S_2 then some property P holds for the composed system $S_1 \odot S_2$. The types of properties generally desired are *safety* and *liveness* (the absence of deadlock), although many other useful properties can be imagined. Since we are interested in robotics, for example, we also consider *stability* or robustness against disturbances (see Section 2.1.2).

Two extremes of the approach can be distinguished, having to do with how strict the requirements on components are. At one extreme, which one might call the *gen-*

eral approach, one: (1) defines a composition of two components to be a certain transformation on their combined states and actions and (2) proves, given two particular components, that their composition has certain properties. Thus, the burden is on the designer to prove that the systems she has composed behave correctly. At the other extreme, which one might call the *careful* approach, one can: (1) design a class of components all of which satisfy some property P, and (2) design compositions that only apply to components with property P and always guarantee composed systems with some desired property Q. Thus, each compositional operator requires only an initial proof which may be applied again and again. For example, consider the basic definition of parallel composition used in process algebras [5, 35, 57] wherein two processes A and B with common transition labels are combined into $A||B$, which is a kind of cross product process with common transitions identified. What $A||B$ does depends on what A and B do in isolation and so it may or may not be safe, live and so on. Thus, this composition is a *general* one where extra formal work awaits the designer. In composing specifications, this is often facilitated by only considering fairly weak properties [1]. An example of the *careful* approach is the parallel composition of cyclic processes along a single shared transition guaranteed to eventually fire. By considering only cyclic processes with a single shared transition, one can guarantee that the resulting composition is also cyclic without proving anything extra. In Section 2.2, which summarizes [42, 39, 41], we elaborate on this notion and show how to build, in a compositional manner, a large class of useful cyclic processes — an example of how the method can scale. Another example of the *careful* approach that is proving practical is the annotational technique where the programmer is required to annotate portions of her code with specifications of, for example, the resources it will require [77] or a proof of what it does [17]. A compiler then reasons about what the entire program will (or should) do. These methods are a response to the scalability problems faced by more general approaches to program semantics such as partial correctness proofs [34], which can be intractable.

The proofs that composed systems have certain properties vary depending on the degree to which known properties of the components can be leveraged. This usually depends on the degree to which the component systems *do not* interact. The system obtained from a general composition of the form $A||B$ is essentially a machine describing the *interaction* between A and B; thus, properties of the two components in isolation are not likely to be of much use [49], although if compositions are suitably careful some properties of the components can be used [24]. This is not to say that a composition that does not have a compositional proof is not useful. The point behind composition is to begin with the intuition that a complicated system can be thought of as a combination of simpler systems — and not, in the general approach, to treat composition merely as an aid to verification. Certainly in the careful approach, a composition for a given type of component need be verified only once.

In Section 2.1 of this chapter we review several compositional methods found in the robotics literature. Because the idea is relatively new (in robotics) — and particularly challenging to apply in real world, continuous, uncertain systems — many of these methods are informal. In Section 2.1.2, we turn to the compositions of controlled dynamical systems developed by Koditschek and his group [15, 13, 59, 75, 73, 14, 73] out of which grew the present dissertation. Most of these are general parallel compositions and, as far as we know, a new proof is required for each application. Sequential composition [15] is careful, however, and applies to dynamical systems with known attractor structures and known domains of attraction. In Section 2.2 we review in some depth a method for composing hybrid factory robot programs that is careful and has compositional proofs [42, 39]. It is an extension of sequential composition to concurrent systems.

The main results of this dissertation, presented in Chapters 3 and 4, concern the composition of cyclic dynamical systems. In Chapter 3 we show that any set of cyclic dynamical systems of the form $\dot{\phi} = u$ can be composed into a phase regulated set of systems. This composition is, therefore, careful. In Chapter 4, we define a composition of cyclic intermittent contact systems (those that admit a control authority only in certain states) that has as its basis the composition in Chapter 3. This latter composition is general in the sense that we have not yet characterized the class of *all* intermittent contact systems to which it may be successfully applied. Instead we show that it produces locally stable systems when applied to ball batting and hopping robot systems. We also demonstrate, in simulation, that it seems to be applicable to other intermittent contact systems as well.

2.1 Composition in Robotics

Inspired by the modularity and scalability of compositional methods in computer science, robotics researchers have begun to find ways to use compositional ideas to design increasingly complicated robot behaviors. Unfortunately, compositional robotics is difficult for all the same reasons that robotics in general is difficult: Useful dynamic mechanisms are nonlinear and admit no general control methodology, much less a compositional one; composed components interact not only logically, but physically too, via difficult to model mechanical and sensory pathways; often the environment is unstructured and uncertain so that the semantics of components is poorly defined; and so on. Due to these problems, compositional methods in robotics are either informal — used as programmer's aids and verified through experiment or simulation — or they are applied only in very structured and constrained environments. This section is a review of some of the approaches taken with an emphasis, in Section 2.1.2, on the composition of robot controllers in fairly structured, mechanically dynamic environments.

2.1.1 Review of Approaches

Many intuitive approaches to the composition of robot behaviors are inspired by Brooks' subsumption architecture [10], which is essentially as design methodology. In this layered approach, a lowest layer consists of basic behaviors (e.g., the application of a voltage to a motor). Subsequent layers inhibit or activate the lower layers according to their own rules. With creativity and luck, a useful behavior "emerges" from the system when it is combined with its environment. Thought of compositionally, each behavior defines a component, and the interfaces between components (patterns of inhibition and activation) define compositions. The resulting architecture provides some modularity and insight to the designer, although no formal results or verification against specifications are likely to emerge from this method [11].

A similar but higher level approach is taken by Arkin [3] to create reactive behavior programs for mobile robots in unstructured environments. A number of low level behaviors, called *schema*, provide the basic components. Schema such as "Move Ahead" or "Dodge" are represented by vector fields on the configuration space of a robot. A *schema manager* then, according to rules set by the designer, superimposes a selection of schema to make more complex behaviors. One commonly used kind of schema essentially moves the robot in random directions. The schema manager uses a schema in this class when it senses that no progress is being made. The frequent use of such random behaviors may be an indication of the trial and error nature of the whole approach, although it may also be the only way to deal with unstructured environments. Extending this idea, other researchers [50, 76] have begun to develop the means to express more subtle combinations of so called "dynamical systems" controllers.

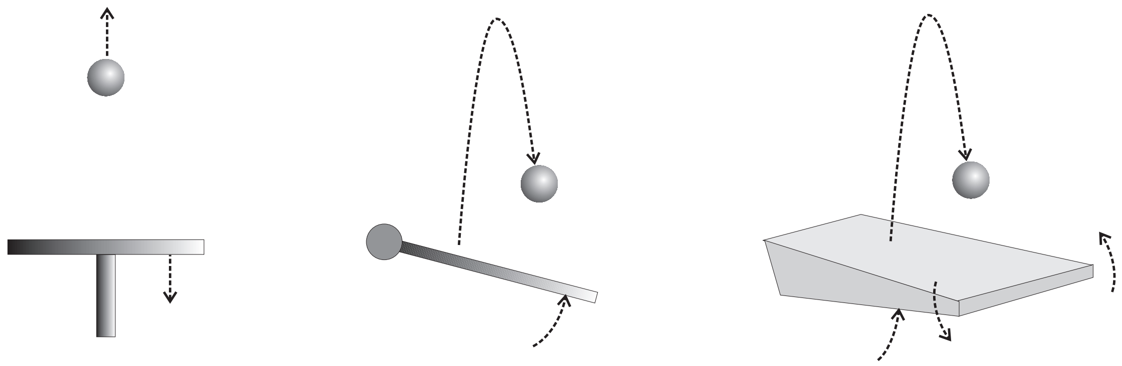
There is some biological basis for superimposing vector field controllers, although the evidence is subject to interpretation. Stimulating different points on the spinal chord of a frog, researchers [7, 58] observe that the muscles produce different force fields at the ankle of the animal. When multiple points are stimulated, the corresponding force fields are apparently superimposed. It is unclear how this fact can be used to deduce how the frog executes locomotor behaviors. It has nevertheless served as partial inspiration for a formal treatment of combining vector fields (in this and other ways) called contraction map analysis [53]. Contraction maps form a class of nonlinear dynamical systems that is preserved under superposition and parallelization. Contraction maps admit a certain level of analysis and have applications in control. They in principle allow the designer to compose new contraction maps from old ones. However, the property that a composed system is contracting if and only if its components are contracting may not imply any useful behaviors beyond an amenity to analysis.

An approach to composition that begins to look more like traditional computer science composition is based on discrete event system composition [48]. Here basic, low level behaviors are represented by potential field controllers [70, 38] contained in simple discrete event supervisors [18]. The supervisors are then composed in various ways with semantics similar to composition in process algebra [5]. The syntax for these programs is tantalizingly similar, on the surface, to the syntax one imagines a robot programming language would have. However, although the feedback from the environment is given by events, a model of the source of these events is missing, possibly requiring the use of a theory of partial knowledge to give it a valid semantics beyond the relatively simple semantics of the finite state machines that describe the supervisory controllers.

All of the approaches mentioned above are attempts to devise a means of composing complex behaviors from simpler ones. They are more or less successful not only to the extent that they allow the designer to express desired behaviors, but also insofar as they provide a formalism for reasoning about behaviors. Clearly the former area has seen more success than has the latter. In the next section we review approaches to composing control algorithms that begin to provide such a formalism.

2.1.2 Composition of Juggling Behaviors

While the approaches to composition in Section 2.1.1 may be appropriate for highly kinematic environments (where the natural dynamics of the machines studied are often highly damped), they are not usable in situations where an exchange of energy between robot and environment is required (e.g., in the dynamical manipulation of objects [72], or in running and hopping [66, 46, 32]). These situations seem to demand the language of dynamical systems for their encoding. In this section we review the approach of Koditschek and his collaborators toward composing dynamical controllers for locomotion and nonprehensile manipulation (ball batting) tasks. Much of this work is inspired by the pioneering and spectacular work of Raibert [67] in robot hopping. For example, controllers for stabilizing the body pitch, hopping height and forward velocity of a planar hopping robot, all conceived of in isolation, are run simultaneously as though they were decoupled [66]. Possibly because each controller itself is a robust, feedback controller and because (luckily) the three tasks are nearly decoupled, the result is a robot that performs all three tasks simultaneously. This tactic begins to fail when enough controllers are combined. However, excellent empirical results warrant further investigation and refinement of these techniques to



(a) (b) (c)

Figure 2.1: Successively more complex compositions of limit cycles (vertical orbits of the ball) with point attractors (horizontal positions of the ball). (a) The vertical juggle of one ball on a piston. (b) The planar juggle of one ball on a paddle. (c) The 3D juggle of one ball in space.

enable applications in other domains.

For example, batting a ball on a paddle (Figure 2.1), which has come to be referred to as *juggling* in the robotics literature, is quite similar to hopping. The fundamental control problem in this task is to achieve a stable apex position of the ball and serves as the basic behavior of the compositional methods discussed in this section and in part of Chapter 4. Such a task is similar to hopping [67], where a spring legged robot is controlled to stabilize a certain hopping height by appropriately delivering thrust to the leg at certain times. In [46] a simplified, one degree of freedom hopper is studied, and a simple control mechanism (delivering thrust for a constant period of time at the point of maximal compression of the leg) is shown to stabilize a limit cyclic in the position/velocity plane.[1]

In juggling, as we have said, the central task is also to stabilize to a limit cycle in the vertical position/vertical velocity of the ball plane. Several increasingly complex versions of this problem are examined in [13] and [73], as illustrated in Figure 2.1. In the most simple version, the ball is constrained to move only vertically and is batted by a piston, Figure 2.1(a). The controller for the piston tracks a reference trajectory

[1] A similar model, in which the stiffness of the leg-spring can be changed at the point of maximal compression, is discussed in Chapter 4.

obtained by "mirroring" the ball's position

$$r = -\mu(b, \dot{b})b \tag{2.1}$$

where r is the position of the robot, b is the position of the ball and μ is a positive function of the state of the ball (defined in terms of its energy so that it is essentially constant between collisions with the paddle). A suitable definition of μ results in a *mirror law* that stabilizes the ball at a particular height. The next most complex version of the task is to bat the ball in the plane, requiring the stabilization of not only the vertical apex of the ball, but also the horizontal position of the ball (Figure 2.1(b)). To achieve this, this basic mirror law controller is composed with a PD style controller for the horizontal stabilization task [44]. The resulting reference trajectory for the robot is then

$$\theta = -\mu(b, \dot{b})\theta_{ball} + h(b, \dot{b}) \tag{2.2}$$

where θ is the angle of the robot paddle, θ_{ball} is the "angle" of the ball with respect to the center of rotation of the paddle, and h is the PD style controller for horizontal stabilization. Both μ and h contain constant gains which must be tuned to stabilize the controller so that (2.2) is essentially the weighted sum of two controllers. Both (2.1) and (2.2) can be shown to be locally stable [13, 12]. The task of juggling in three dimensions with a three jointed robot paddle (Figure 2.1(c)) is addressed in [73] where the vertical height of the ball and the now two dimensional horizontal state of the ball are stabilized via the composition of a mirror law controller and a two degree of freedom PD style controller. The analysis of this controller is addressed in [72].

The composition of mirror law controllers with PD style controllers (by summing them) is *general* in the sense coined in the beginning of this chapter. That is, since the class of basic tasks to which the composition can be successfully applied has not been determined, each new composition of controllers must be shown to be correct (stable) in each new situation. Furthermore, essentially no information about the stability of the controllers in isolation is used in the proofs — although this information does guide the intuition of the system designer. Rather a certain invariant manifold in the configuration space of the ball is shown to be locally stable using a distinctly noncompositional analysis. Nevertheless, this approach to the parallel composition of cycles with point attractors can claim some degree of generality. For example, the controller for a functioning two degree of freedom brachiating robot [59] may be interpreted as the parallel composition of a repetitively swung simple pendulum with a nonlinear PD. Work in progress suggests that similar parallel compositions of a repetitively swung pogo stick with a nonlinear PD may yield effective controllers for multi-jointed [75] and even multi-legged [2] running. These last examples, moreover, begin to hint at a hypothetical explanation of how running animals exhibit ground reaction forces characteristic of pogo sticks [29]. But for none of these examples has a general and formal coupling procedure been articulated: a compositional "recipe" common to the batters, the brachiator and the runners is not yet evident.

Compositions of the sort described above, involving point attractors or points and one cyclic attractor, satisfy the following steady state behavior: The attractor of the coupled system is the cross product of the component attractors. The obvious next step is to consider the composition of multiple limit cycles, that is, to *juggle* several balls at once. This idea was introduced by Bühler and Koditschek for a planar robot [14] and extended to a spatial juggler by Rizzi and Koditschek [72]. In each case, a continuous *switch* was used to alternate between mirror laws for each ball. The construction of such a switch is disscussed in Appendix A. A formal treatment of these ideas is given in this thesis in Chapter 4.

A simpler composition of controllers, used in juggling and elsewhere, is also probably the most obvious kind of composition. It is the *sequential* composition of controllers. Given two controllers, the goal of the first being in the domain of the second, we simply run one after the other. Sequential composition, also known as preimage backchaining, was introduced into the motion planning literature in [55] as a method of sequentially composing motion strategies. In [15] this method was extended to dynamically dexterous robot manipulators. The idea is to start with a palette of controllers $\Phi_1, ..., \Phi_n$ for the motion of a robot with configuration space X. Suppose Φ_k has domain $\mathcal{D}_k$ and goal $\mathcal{G}_k$. Order the palette by setting $\Phi_i \sqsupseteq \Phi_j$ (read Φ_i *prepares* Φ_j) whenever $\mathcal{G}_i \subseteq \mathcal{D}_j$. If the palette is suitably designed, then a switching strategy may be obtained that drives the robot to a goal from any initial condition in the union of the domains in the palette, thereby constructing a possibly quite large domain from the possibly small domains of otherwise robust controllers. The combination of all the Φ_k's forms a new controller. Sequential composition is the basis of the concurrent composition reviewed in Section 2.2 where the idea is expanded to include concurrent, decentralized motion control.

2.2 An Example of Scalability: Threaded Petri Nets

In this section we review an approach to composition that extends sequential composition to concurrent systems, described in [42] and [39] and developed by the author of this thesis. The approach is based on a kind of Petri Net [69] called a *Threaded Petri Net* (TPN). We only outline the definitions and results to give a flavor of the power of the careful approach to composition. The inspiration for the approach is the idea of a *bucket brigade*: a line of people passing buckets or sand-bags. A robotic bucket brigade is a perfect example of a decentralized robotic system. Each robot has the same program: "Wait for previous robot; receive bucket; wait for next robot; pass bucket; repeat." A three robot brigade is modeled by a TPN as shown in Figure 2.2. Each of the three cycles represents a copy of the robot program just described. The three cycles are composed together to form the brigade. In particular, each robot only interacts with its neighbors so that the system scales up to arbitrary length brigades. The main application of TPNs to date is the control of decentralized assembly systems in manufacturing and is particularly influenced by the modular approach to factory design exemplified by the *minifactory* [71].

Specifically, a TPN is a simple Petri Net or *marked graph* [23] augmented with a means of assigning and reassigning the (controllable) degrees of freedom of a robotic system to a palette of continuous controllers. Thus, to each *place* in a TPN corresponds a function describing a controlled dynamical system. We therefore call places *controllers*. To each *transition* in a TPN there is a *redistribution function* that reassigns the degrees of freedom in its *preset* to the controllers in its *postset*. A *marking* of a TPN is an assignment of each degree of freedom to a particular *role* in one of

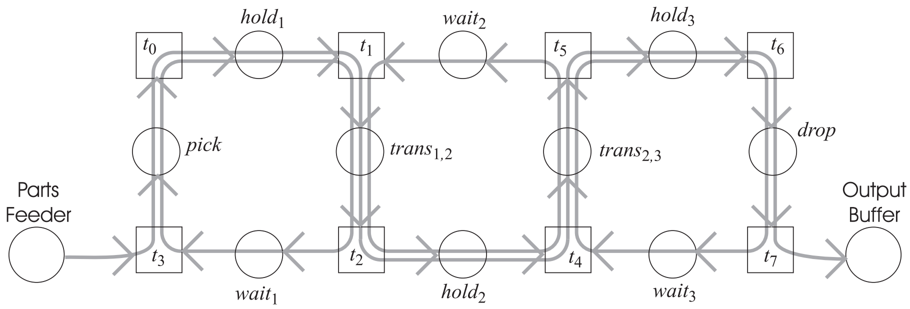


Figure 2.2: Schematic of a TPN Model of a three robot brigade. Gray lines illustrate the redistribution of degrees of freedom among controllers.

the controllers. A transition *fires* when all of the controllers in its preset are in goal states. The mathematical details of TPNs are beyond the scope of this dissertation but can be found in [42, 39] and the proofs of various TPN properties in [41]. For example, in the TPN in Figure 2.2, a marking may assign robot R_1 and a particular bucket B to the controller $hold_1$ and robot R_2 to the controller $wait_2$. When both of these controllers reach goal states — R_1 is firmly holding B and R_2 is waiting to receive a bucket — then transition t_1 fires, reassigning the robots and the bucket to appropriate roles in the controller $trans_{1,2}$, which results in the change of possession of B from R_1 to R_2.

TPNs, therefore, extend sequential composition to *concurrent* composition. Instead of sequencing a set of controllers — each of which operates on all controllable degrees of freedom — where one controller may follow another if its domain of attraction contains the goal set of the other (Section 2.1.2), TPNs use the redistribution function. Roughly, the cross product of the goals in the preset of a transition must be contained in the cross product of the domains of the postset of the transition. This allows many controllers to operate and transition to other controllers simultaneously, taking advantage of the fact that in many manufacturing systems, most pairs of subsystems are decoupled.

Petri Nets themselves can also be composed, by taking unions of nets with common places and transitions [37]. A careful such compositional method, called *gear net* composition [41], is the basis for a "factory compiler" [39]. The point is to compose

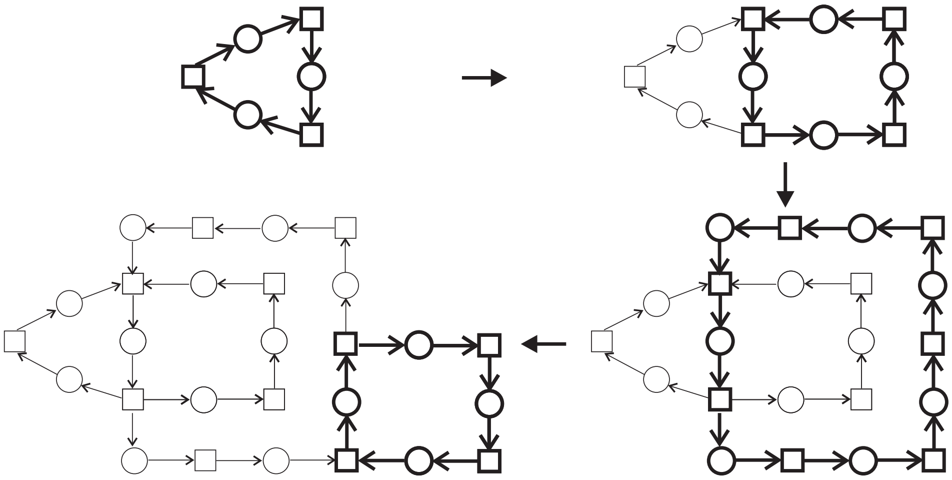


Figure 2.3: The iterative process of constructing a gear net from simple gears.

single-cycle Petri Nets with gear nets in such a way that *liveness* and *reversibility* are preserved. Figure 2.3 illustrates gear net composition. Starting with a single cycle (or gear), a new cycle is attached. The resulting two-cycle net is live and reversible (under the assumption that each cycle is marked exactly once). Note that the attachment site is of the form transition-place-transition. Next, another cycle is attached to the net and so on, always using the same attachment scheme. One of the main theorems concerning TPNs [41] is that nets so composed are *always* live and reversible. Thus, gear net composition, which only applies to gear nets and cycles, is a careful composition.

Based on gear net composition, a "factory compiler" has been built and implemented in a simplified, simulated setting [39]. The compiler takes as input a *product assembly graph*, which describes how a product is obtained from an operation on its subassemblies, how those subassemblies are obtained and so on. The compiler produces as output a description of a factory that produces the product. The factory description includes a topological layout of robot workspaces and programs for each robot to run. The collection of programs can be shown to implement a gear net and is thus live and reversible. Furthermore, for a certain class of product assembly graphs, this is always the case. Experiments with a simulation of these factories suggest that the distributed nature of the programming allows the method to scale well. For ex-

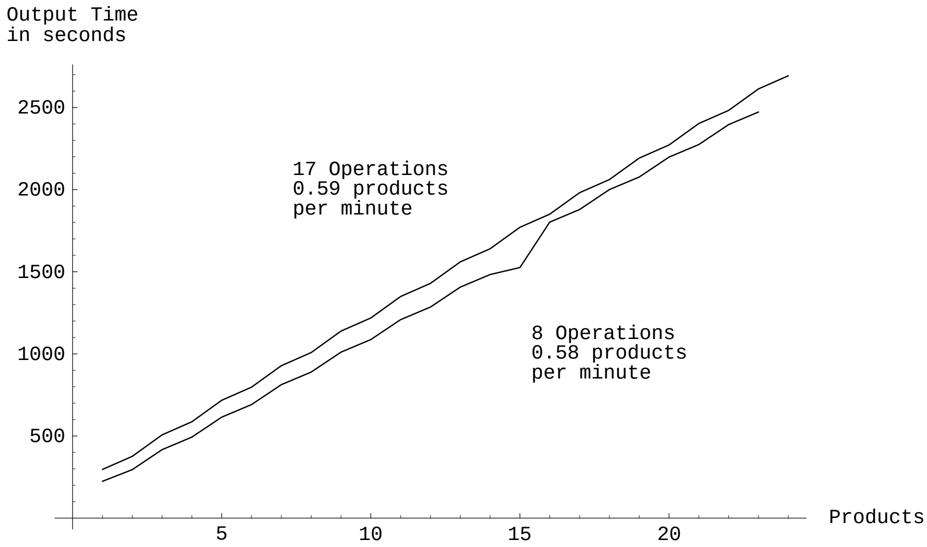


Figure 2.4: Two different factories with 8 and 17 operations respectively show different startup times (vertical offsets) but similar production rates (slopes).

ample, the rate at which products come off the assembly line is independent of the number of operations in the factory. In the simulation, after an initial start up time, during which no products are completed as the initial parts propagate through to the output buffer, products show up at fairly regular intervals: about 0.6 products per simulated minute in all factories tested.

There is a strong cyclic nature to gear net models. This is not surprising, since they are based on gear nets which are cyclic systems. More interesting is that the factories seem to settle into a certain cyclic velocity — the rate at which they produce products — although an analysis of how wait modes propagate backwards from the factory output buffer to the parts feeders has not been done as shown in Figure 2.4. The simple gear may, in the context of a gear-net-shaped TPN, be functioning more like a true oscillator than the basic Petri Net semantics describe. Thus, in the next chapter we will begin to develop an alternative to Petri Nets called *phase regulation.* The examples we can presently provide for phase regulation in Chapters 4 and 5 are much different on the surface from the periodic processes in manufacturing. However, we believe a link may someday be found that allows the use of the much more dynamic and seemingly better suited phase regulation techniques described in Chapter 3 to be used in manufacturing.

CHAPTER 3

Coupling First Order Cyclic Systems

In this chapter we consider the composition of n cyclic robotic systems, each of which has control over its phase velocity (so that $\dot{\theta}_i = u_i$) and may use its own state and the state of its *neighbors* in computing its update rule. We suppose that a task is represented as a limit cycle — an isolated invariant periodic attractor — on a torus of some dimension. For now we suppose that a suitable computationally effective measure of *phase* such as action angel coordinates [4] — an abstraction to this model space from the state space of the robot system in question — can be found. In Chapter 4, in which we consider intermittent contact systems, we show how to construct phase coordinates for certain types of systems. We also assume that all degrees of freedom are actuated. Later, also in Chapter 4, we relax this assumption. We call the vector fields corresponding to the model dynamical systems we construct *reference fields*, roughly comparable to the target dynamics [59] or templates [29] introduced in allied papers. In Chapter 4 we use these fields to construct feedback controllers that attempt to make systems with less direct control over their phase velocities behave like the model systems we construct by having them *refer* to the value of the field at each point in their configuration space.

Given a configuration space X with dynamics $\dot{\mathbf{x}} = F(\mathbf{x}, \mathbf{u})$, we define a **task** for this system to be a submanifold $M \subseteq X$ with a dynamical system of the form $\dot{\mathbf{y}} = G(\mathbf{y}) \in T_{\mathbf{y}}M$ defined over it. A control law $\mathbf{u} = \mathbf{g}(\mathbf{x})$ **performs** the task (M, G) if

1. M is an attractor (i.e., an asymptotically stable invariant set) and

2. the restriction dynamics on M is given by G.

In this chapter, we wish to control n one dimensional oscillators each of whose phase $\phi_i(t)$ can be thought of as a point in the circle S^1. We take as a representation of the circle the interval $[0, 2\pi]$ with its end points identified. Thus, all values of ϕ_i are considered modulo 2π. We frequently consider smooth 2π-periodic functions of $\mathbb{R}$ and smooth functions on S^1 equivalently. Each oscillator also has a phase velocity $\dot{\phi}(t) \in \mathbb{R}$ which, as discussed in Section 1, we consider to be the control input for the oscillator. Thus, $\dot{\phi}_i = u_i$.

The configuration space of n such oscillators is the n-dimensional torus,

$$\mathbb{T}^n = \underbrace{S^1 \times ... \times S^1}_{n \text{ times}}.$$

The tasks over the torus that we consider in this chapter are constant flows on subtori. They thus have the form $(\mathbb{T}^m, G)$ where $0 < m \leq n$ and for $\theta \in \mathbb{T}^m$, $G(\theta) = \mathbf{a}$, $\mathbf{a} \in \mathbb{R}^m$. Since we are presently only interested in limit cycles, we take $m = 1$ so that $\mathbb{T}^m = S^1$. In Section 3.2 we show how to construct reference fields over $\mathbb{T}^2$, and in Section 3.3 we extend the idea to a particular task in n dimensions: the problem of coordinating the oscillators so that they exhibit particular phase relationships in a decentralized manner.

3.1 Specifying Phase Relationships

As mentioned in Section 2.1.2, composing some number of cyclic tasks requires a specification of the desired phase relationships between them. We also would like to specify the communication structure to be used by the system by stipulating which oscillators are neighbors of any particular oscillator. To proceed, we define a *connection graph* as follows. Define C to be an $n \times n$ symmetric matrix over the set $\{0, 1, -1\}$ where $C_{i,i} = 0$ for all i. We interpret C as follows. If $C_{i,j} = 1$, then it is desired that oscillators i and j be in phase: $\phi_i - \phi_j = 0 \pmod{2\pi}$. If $C_{i,j} = -1$, then it is desired that i and j be out of phase: $\phi_i - \phi_j = \pi \pmod{2\pi}$. If $C_{i,j} = 0$ then no phase difference is specified — although one may be implied transitively via other connections.

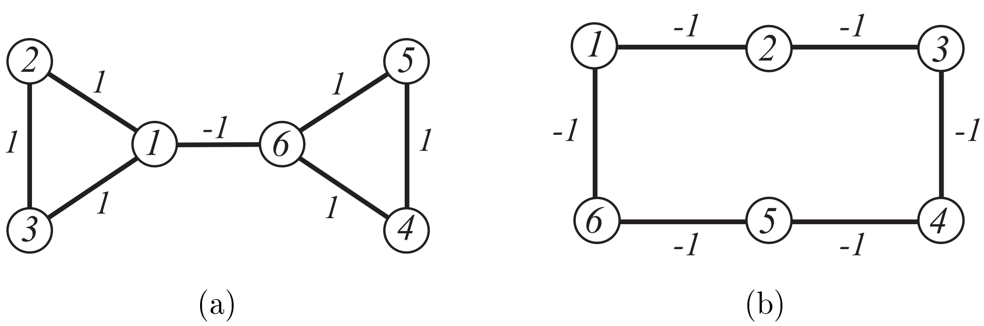


Figure 3.1: Two different connection graphs. The system (3.10) obtained from (a) results in an "alternating tripod" behavior, whereas the system (3.11) obtained from (b) does not.

We describe two examples. The first is a specification suitable as the basis for the control of a six legged robot. It consists of six oscillators, one for each leg, connected so that there are two disjoint, fully connected in-phase tripods and one out-of-phase connection between a representative from each tripod. The second is a specification that produces an unintended behavior — that is, one where the connection matrix does not give a system that performs the task associated with it over the *entire* domain of the system. In Section 3.4 we demonstrate how to check these systems using the results in Section 3.3.

Let $\phi_1, ..., \phi_6$ represent the phases of each of six oscillators and suppose that oscillators one through three correspond to the first tripod and four through six, the second. And suppose one and six are connected out of phase. The connection matrix that realizes this scheme, illustrated in Figure 3.1(a), is

$$C_{alt} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & -1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ -1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}. \tag{3.1}$$

Another seemingly natural way to specify an alternating tripod is to suppose that the six legs are arranged in a ring, with each out of phase with its neighbors, as illustrated in Figure 3.1(b). It turns out that this specification cannot be realized with our method, as is shown in Section 3.4.

$$C_{bad} = \begin{pmatrix} 0 & -1 & 0 & 0 & 0 & -1 \\ -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 \\ -1 & 0 & 0 & 0 & -1 & 0 \end{pmatrix} . \tag{3.2}$$

3.2 Two Dimensions

To illustrate our idea, we consider the task of regulating two oscillators with phases ϕ_1 and ϕ_2 so that (1) the rate of change of each phase is some desired value and (2) the phases are maximally separated. This is specified by the connection matrix

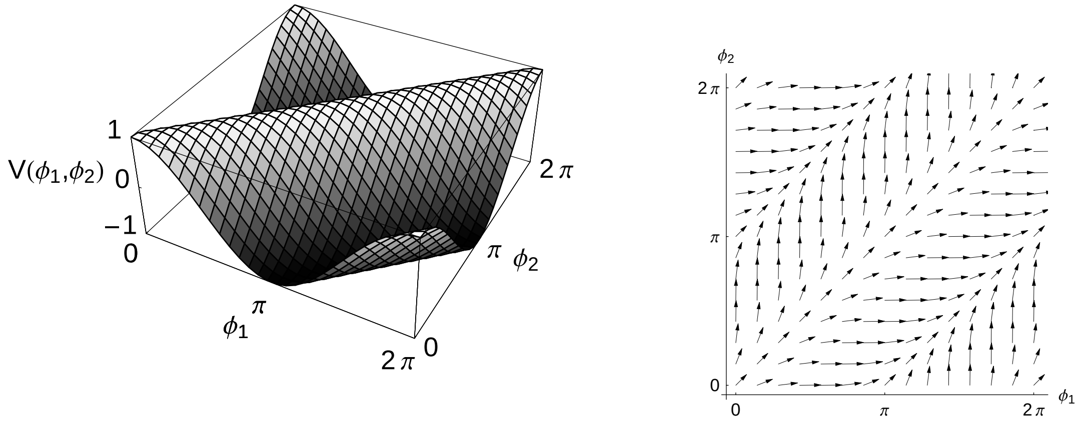
$$C_2 = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} . \tag{3.3}$$

To define a reference system for this task, we proceed in two steps. First, we define a potential energy function V on $\mathbb{T}^2$ which has a unique minimum at $\phi_2 = \phi_1 + \pi \pmod{2\pi}$. Then we take the negative gradient $-\nabla V$ and add a *drift* term $(1,1)^T$ which *suspends* the gradient system in the two dimensional torus. We define the potential energy function V over $\mathbb{T}^2$ by $V(\phi_1, \phi_2) = \cos(\phi_2 - \phi_1)$

This function, shown in Figure 3.2(a), has the set $\{(\phi_1, \phi_2) \mid \phi_1 - \phi_2 = \pi \pmod{2\pi}\}$ as its minimum. We define the reference field to be

$$\mathcal{R}(\phi_1, \phi_2)^T = \kappa_1 \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \kappa_2 \nabla V(\phi_1, \phi_2) . \tag{3.4}$$

Here κ_2 is an adjustable gain that controls the rate of convergence to the limit cycle. The circles $\phi_2 = \phi_1$ and $\phi_2 = \phi_1 + \pi$ are equilibrium orbits. The first is unstable, the second is stable. See Figure 3.2(b). Thus, this field performs the task specified by C_2. Speaking compositionally, we say that the two separate tasks $\dot{\phi}_1 = \kappa_1$ and $\dot{\phi}_2 = \kappa_1$ have been composed by adding to the system the coupling term $-\kappa_2 \nabla V(\phi_1, \phi_2)$.



(a) (b)

Figure 3.2: The two dimensional reference field. (a) The artificial energy function V on the phase difference $\phi_1 - \phi_2$. (b) The reference field (3.4) obtained by suspending the negative gradient of V in the torus.

3.3 Multiple Oscillators and Arbitrary Connections

In this section we consider multiple oscillators and the further constraint that the reference fields we construct be in a certain sense decentralized. In particular, we build systems $\dot{\mathbf{x}} = \mathcal{R}(\mathbf{x})$ such that $\mathcal{R}_i$ depends only on a subset of the phases in the system — a set of "neighbors" that has been designated by the designer. We examine only systems whose task dynamics have $\dot{\phi}_i = a \in \mathbb{R}$ for all i and concentrate on the decentralized aspect of the problem. In particular, we propose the following method.

1. Identify the *connections* desired in the reference field. That is, specify with which neighbors the controller of a given oscillator must communicate.

2. Label the connections as either "in phase" or "out of phase."

3. Construct an energy function that respects these constraints.

4. Suspend (a generalized version of) the gradient of this function to complete the reference field and check that it meets the specifications given by the labeling.

The last step arises because the third step may not be able to realize the phase relationships specified in the second step while respecting the connections enforced in the first step. However, we supply a criterion to check this property.

Definition 3.3.1 *The* **task specified** *by a given connection matrix C is given by the set*

$$M_C = \{(\phi_1, ..., \phi_n) \mid \forall i, j\ C_{i,j} \neq 0 \to \phi_i - \phi_j = \frac{\pi}{2}(1 - C_{i,j})\}$$

with the dynamics

$$(\dot{\phi}_1, ..., \dot{\phi}_n) = \kappa_1(1, ..., 1)\ .$$

We will give a reference field $\mathcal{R}$ based on C that performs this task in some instances, depending on the structure of C. As we have stated, however, we supply a criterion to check that a given reference field actually performs the specified task.

From C we define a reference field $\mathcal{R}_C(\phi_1, ..., \phi_n)$ by setting

$$\dot{\phi}_i = \kappa_1 - \kappa_2 \sum_{j=1}^{n} C_{i,j} \sin(\phi_i - \phi_j) \tag{3.5}$$

where κ_1 and κ_2 are constant gains. We will show that $\mathcal{R}_C$ arises from the suspension of a gradient-like system on the subtorus defined on the differences between the oscillators. In particular, let $V_{i,j} = -C_{i,j}\cos(\phi_i - \phi_j)$ and set V to be the energy function

$$V = \sum_{i<j} V_{i,j}\ . \tag{3.6}$$

To simplify our analysis, we will transform (3.5) to a more convenient form. To this end we introduce the following notation and definitions.

Let $\mathbf{x} = (\phi_1, ..., \phi_n)^T$. We will first define the system $\mathbf{y} = L_n\mathbf{x}$ obtained from the phases of the oscillators to be all possible differences between phases, corresponding to the connections in the graph.

Definition 3.3.2 *For each $n > 1$ we define the $\frac{n(n-1)}{2} \times n$ dimensional* **difference matrix** *L_n recursively as follows. First, set $L_2 = (1, -1)$. Then, for each $n > 2$ set*

$$L_n = \left(\begin{array}{c|c} \mathbf{1} & -I_{n-1} \\ \hline \mathbf{0} & L_{n-1} \end{array}\right)$$

where I_n is the $n \times n$ identity matrix, the upper left hand part of the matrix is an $(n-1) \times 1$ vector of ones and the lower left hand part of the matrix is an $\frac{(n-1)(n-2)}{2} \times 1$ vector of zeros.

For example, with four oscillators, the difference matrix is

$$L_4 = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

and $\mathbf{y} = L_4\mathbf{x} = (x_1 - x_2, x_1 - x_3, x_1 - x_4, x_2 - x_3, x_2 - x_4, x_3 - x_4)^T$.

We now express the reference field (3.5) in this notation.

Definition 3.3.3 *Suppose the $n \times n$ dimensional connection matrix C is given. Let $\tilde{C}$ be the $\frac{n(n-1)}{2} \times \frac{n(n-1)}{2}$ dimensional diagonal matrix*

$$\tilde{C} = \begin{pmatrix} C_{1,2} & & & & & & & \\ & \ddots & & & & & & \\ & & C_{1,n} & & & & & \\ & & & C_{2,3} & & & & \\ & & & & \ddots & & & \\ & & & & & C_{2,n} & & \\ & & & & & & \ddots & \\ & & & & & & & C_{n-1,n} \end{pmatrix}$$

Also, let $\mathbf{1} = (1, ..., 1)^T$ be a vector of n ones. The matrix $\tilde{C}$ essentially lists the edges of C down its diagonal. Then the **reference field** *associated with C is*

$$\mathcal{R}_C(\mathbf{x}) = \kappa_1 \mathbf{1} - \kappa_2 L_n^T \tilde{C} s(L_n \mathbf{x}) \tag{3.7}$$

where $\mathbf{x} = (\phi_1, ..., \phi_n)^T$ and $s(\mathbf{y}) = (\sin(y_1), ..., \sin(y_m))^T$.

The reference field R_C induces a vector field which we denote by G on the connections $\mathbf{y} = L_n\mathbf{x}$ given by

$$\begin{aligned} \dot{\mathbf{y}} &= G(\mathbf{y}) \triangleq L_n\dot{\mathbf{x}} = \kappa_1 L_n \mathbf{1} - \kappa_2 L_n L_n^T \tilde{C} s(\mathbf{y}) \\ &= -\kappa_2 L_n L_n^T \tilde{C} s(\mathbf{y}) \end{aligned} \tag{3.8}$$

since $L_n\mathbf{1} = 0$.

To understand the dynamics of (3.8), and therefore of R_C, we show that they are gradient-like. That is, its equilibrium states are the minima of some energy like-function. A function $U : X \to \mathbb{R}$ is a *LaSalle Function* (a generalized Lyapunov function) for the vector field $\dot{x} = F(x)$ if its image is compact and if $\dot{U}(x) \triangleq DU \cdot F(x) \leq 0$. We have

Lemma 3.3.1 *The energy function $V : \mathbb{R}^{\frac{n(n-1)}{2}} \to \mathbb{R}$ defined by (3.6) is a LaSalle function for G.*

Proof: Since $V(\mathbf{y}) = -\sum_{i<j} C_{i,j}\cos(y_{i,j})$, taking derivatives gives $DV = s(\mathbf{y})^T\tilde{C}$. Then

$$DV \cdot G(\mathbf{y}) = -\kappa_2 s(\mathbf{y})^T\tilde{C}L_nL_n^T\tilde{C}s(\mathbf{y}) = -\kappa_2||L_n^T\tilde{C}s(\mathbf{y})|| \leq 0$$

since $\tilde{C}$ is symmetric. Finally, the image of V is contained in $[-\frac{n(n-1)}{2}, \frac{n(n-1)}{2}]$, proving the lemma. $\Box$

Let $\omega(Y)$ be the forward limit set of the set $Y \triangleq \mathbb{R}^{\frac{n(n-1)}{2}}$ under the dynamical system $\dot{\mathbf{y}} = G(\mathbf{y})$. Our first result is a characterization of $\omega(Y)$. We have

Theorem 3.3.1 *The forward limit set $\omega(Y)$ of G is equal to $\{\mathbf{y} \mid G(\mathbf{y}) = 0\}$, the zeros of G.*

This is the criterion for checking that the specifications given by C are met by R_C — we will discuss this after we prove the theorem. First we note a simple fact:
Fact: If $L_n\mathbf{w} = 0$ then $w_1 = w_2 = ... = w_n$. This follows from the form of L_n in Definition 3.3.2. If $L_n\mathbf{w} = 0$ then

$$L_n\mathbf{w} = \left(\begin{array}{c|c} \mathbf{1} & -I_{n-1} \\ \hline \mathbf{0} & L_{n-1} \end{array}\right)\begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_1 \end{pmatrix} - I_{n-1}\begin{pmatrix} w_2 \\ \vdots \\ w_n \end{pmatrix} \\ L_{n-1}\begin{pmatrix} w_2 \\ \vdots \\ w_n \end{pmatrix} \end{pmatrix} = \mathbf{0},$$

from which the result is obvious.

We also recall LaSalle's Invariance Principle [51], which is used in the proof of Theorem 3.3.1.

Theorem 3.3.2 *(LaSalle's Invariance Principle) If $V : Y \to \mathbb{R}$ is a LaSalle function for G, then $\omega(Y) \subseteq \{y \mid \dot{V}(y) = 0\}$.*

Proof (of Theorem 3.3.1)**:** We will first show that $L_n L_n^T \mathbf{y} = 0$ if and only if $L_n^T \mathbf{y} = 0$. The result then follows from set equalities. So suppose that $L_n L_n^T \mathbf{y} = 0$. Let $\mathbf{w} = L_n^T y$. Then $L_n \mathbf{w} = 0$ implies that $w_1 = w_2$, $w_1 = w_3$, ..., $w_{n-1} = w_n$ by the above fact. Thus, $\mathbf{w} = L_n^T \mathbf{y} = \alpha \mathbf{1}$ for some $\alpha \in \mathbb{R}$. Also, taking the transpose of $L_n \mathbf{1} = 0$ gives $\mathbf{1}^T L_n^T = 0$ from which we can conclude that $\mathbf{1}^T L_n^T \mathbf{y} = 0$. This means that $\mathbf{1}^T \alpha \mathbf{1} = 0$ which implies that $\alpha = 0$. Thus, $L^T \mathbf{y} = 0$. We conclude that $LL^T \mathbf{y} = 0 \Leftrightarrow L^T \mathbf{y} = 0$.

Now we have

$$\begin{aligned}
\omega(Y) &\subseteq \{\mathbf{y} \mid \dot{V}(\mathbf{y}) = 0\} \\
&= \{\mathbf{y} \mid s(\mathbf{y})^T \tilde{C} L_n L_n^T \tilde{C} s(\mathbf{y}) = 0\} \\
&= \{\mathbf{y} \mid L^T \tilde{C} s(\mathbf{y}) = 0\} \\
&= \{\mathbf{y} \mid LL^T \tilde{C} s(\mathbf{y}) = 0\} \\
&= \{\mathbf{y} \mid G(\mathbf{y}) = 0\} .
\end{aligned}$$

The first inclusion is by 3.3.1 and LaSalle's Invariance Principle. The second to last equality is because $L_n L_n^T \mathbf{y} = 0 \Leftrightarrow L_n^T \mathbf{y} = 0$. Finally, by definition, $\{\mathbf{y} \mid G(\mathbf{y}) = 0\} \subseteq \omega(Y)$ and thus, $\{\mathbf{y} \mid G(\mathbf{y}) = 0\} = \omega(Y)$. $\square$

Note that the $\dot{\mathbf{y}}$ system has dimension $n(n-1)/2$, yet the differences between n oscillators in a connected network can be characterized by $n-1$ appropriately chosen differences. In other words, the $\dot{\mathbf{y}}$ system has insufficient rank. Thus, we work with the smaller system obtained by taking the first $n-1$ elements of $\mathbf{y}$, to analyze particular systems. From these differences, all other differences may be defined as long as the graph associated with the matrix C is connected. To this end, we define a projection

$$P_n = \left(\begin{array}{c|c} I_{n-1} & 0 \end{array} \right)$$

where the right part is an $\frac{(n-1)(n-2)}{2} \times (n-1)$ dimensional matrix. Then set $\mathbf{z} \triangleq P_n \mathbf{y} = (\phi_1 - \phi_2, \phi_1 - \phi_3, ..., \phi_1 - \phi_n)^T$. We will also need a pseudoinverse of P_n, namely

$$P_n^\dagger = \left(\frac{I_{n-1}}{L_{n-1}} \right).$$

Then the $\mathbf{z}$ system is

$$\dot{\mathbf{z}} = H(\mathbf{z}) \triangleq PG(\mathbf{y}) = \kappa_2 P_n L_n L_n^T \tilde{C} s(P_n^\dagger \mathbf{z}). \tag{3.9}$$

Note that $\tilde{V} \triangleq V \circ P_n^\dagger$ is a LaSalle function for this system.

From the dynamical system defined on the connections between the oscillators, we can deduce the behavior of the total system $\dot{\mathbf{x}} = R_C(\mathbf{x})$. We have

Corollary 3.3.1 *The limit set $\omega(\mathbb{T}^n)$ of the system $\dot{\mathbf{x}} = R_C(\mathbf{x})$ is equal to the set $\{\mathbf{x} \mid H(P_n L_n \mathbf{x}) = 0\}$. The dynamical system restricted to this set is simply $\dot{x}_i = \kappa_1$ for each i. Furthermore, if $\mathbf{z}^*$ is a stable fixed point of H, then the set $\{\mathbf{x} \mid P_n L_n \mathbf{x} = \mathbf{z}^*\}$ is a stable orbit of R_C.*

Thus, to check that the reference field R_C performs the task specified by C, we check that the stable orbits of $\dot{\mathbf{x}} = R_C(\mathbf{x})$, which are given by the stable fixed points of $\dot{\mathbf{z}} = H(\mathbf{z})$, correspond to the task.

3.4 Examples

In this section we apply the above criterion to the examples in Section 3.1.

3.4.1 Example 1: An Alternating Tripod

Start with the connect graph C_{alt} defined in (3.1) and let $\mathbf{x} = (\phi_1, ..., \phi_6)^T$. The system is then

$$\dot{\mathbf{x}} = \kappa_1 \mathbf{1} - \kappa_2 L_6^T \tilde{C}_{alt} s(L_6 \mathbf{x})$$

where $\tilde{C}_{alt}$ is defined as in Definition 3.3.3. To understand this system, we use Theorem 3.3.1 and examine the system (3.8), given in this case by

$$\dot{\mathbf{y}} = -L_6 L_6^T \tilde{C}_{alt} s(\mathbf{y})$$

where $\mathbf{y} = L_6\mathbf{x}$ (without loss of generality we let $\kappa_2 = 1$). Now take the projection $\mathbf{z} \triangleq P_6\mathbf{y}$ to the first five elements of $\mathbf{y}$ to get the system

$$\dot{\mathbf{z}} = H(\mathbf{z}) \triangleq \begin{pmatrix} 2\sin(z_1) + \sin(z_1 - z_2) + \sin(z_2) - \sin(z_5) \\ \sin(z_1) - \sin(z_1 - z_2) + 2\sin(z_2) - \sin(z_5) \\ \sin(z_1) + \sin(z_2) + \sin(z_3 - z_4) + \sin(z_3 - z_5) - \sin(z_5) \\ \sin(z_1) + \sin(z_2) - \sin(z_3 - z_4) + \sin(z_4 - z_5) - \sin(z_5) \\ \sin(z_1) + \sin(z_2) - \sin(z_3 - z_4) - \sin(z_4 - z_5) - 2\sin(z_5) \end{pmatrix}. \quad (3.10)$$

Setting $\dot{\mathbf{z}} = 0$ and solving for $\mathbf{z}$ gives 72 fixed points. Straightforward calculation of the eigenvalues of the Jacobian of H at each of these points shows that only one fixed point, namely $(0, 0, \pi, \pi, \pi)$ is stable and the rest are unstable. Using Corollary 3.3.1, we conclude that the task performed by this system is given by

$$M_{alt} = \{(\phi_1, ..., \phi_6) \mid \phi_1 - \phi_2 = \phi_1 - \phi_3 = 0 \text{ and } \phi_1 - \phi_4 = \phi_1 - \phi_5 = \phi_1 - \phi_6 = \pi\}$$

with the task dynamics $\dot{\phi}_i = \kappa_1$, which is equivalent to the task specified by the connection matrix C_{alt}.

3.4.2 Example 2: An Unintended Behavior

Now consider the specification given by C_{bad} and defined by (3.2). In this case, $\dot{\mathbf{z}} = -P_6 L_6 L_6^T \tilde{C}_{bad} s(P_6^\dagger \mathbf{z})$ is given by

$$\dot{\mathbf{z}} = H(\mathbf{z}) \triangleq \begin{pmatrix} 2\sin(z_1) + \sin(z_1 - z_2) + \sin(z_5) \\ \sin(z_1) - \sin(z_1 - z_2) + \sin(z_2 - z_3) + \sin(z_5) \\ \sin(z_1) - \sin(z_2 - z_3) + \sin(z_3 - z_4) + \sin(z_5) \\ \sin(z_1) - \sin(z_3 - z_4) + \sin(z_4 - z_5) + \sin(z_5) \\ \sin(z_1) - \sin(z_4 - z_5) + 2\sin(z_5) \end{pmatrix}. \quad (3.11)$$

The desired fixed point of this equation is $(\pi, 0, \pi, 0, \pi)$, which is indeed stable. However, if we check the eigenvalues of the Jacobian of H at the point $\mathbf{z}^* = (\frac{2\pi}{3}, \frac{4\pi}{3}, 0, \frac{2\pi}{3}, \frac{4\pi}{3})$, we find that the $\mathbf{z}^*$ is stable, among others. Thus, the system (3.11) does not perform the task specified by C_{bad} because it has five distinct, stable limiting behaviors.

3.4.3 Example 3: Where Linearization Doesn't Work

Next we consider the system, similar to the one in Example 3.4.2, obtained from the four node cycle

$$C_4 = \begin{pmatrix} 0 & -1 & 0 & -1 \\ -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 \\ -1 & 0 & -1 & 0 \end{pmatrix}$$

which gives the $\mathbf{z}$ system

$$\dot{\mathbf{z}} = H(\mathbf{z}) \triangleq \begin{pmatrix} 2\sin(z_1) + \sin(z_1 - z_2) + \sin(z_3) \\ \sin(z_1) - \sin(z_1 - z_2) + \sin(z_2 - z_3) + \sin(z_3) \\ \sin(z_1) - \sin(z_2 - z_3) + 2\sin(z_3) \end{pmatrix}.$$

This system does indeed perform the task associated with C_4. However, the analysis is more difficult than the previous two examples because the solutions to $H(\mathbf{z}) = \mathbf{0}$ are not all isolated, hyperbolic fixed points. In fact, all points in the circle $\{(\pi - s, \pi, s) \mid s \in [0, 2\pi]\}$ are solutions to $H(\mathbf{z}) = \mathbf{0}$ among others. The eigenvalues of the Jacobian of H evaluated at these points are 0, $2\cos(s)$ and $-2\cos(s)$ so we can conclude that all of these points are repelling except $\mathbf{z}_1^* = (\frac{\pi}{2}, 0, \frac{\pi}{2})$ and $\mathbf{z}_2^*(-\frac{\pi}{2}, 0, -\frac{\pi}{2})$. In fact, the Jacobian evaluated at these latter two points evaluates to the zero matrix and so further analysis must be done. Numerically, $\mathbf{z}_1^*$ and $\mathbf{z}_2^*$ do appear to be unstable as do the rest of the zeros of H except for the point $(\pi, 0, \pi)$, which is attracting and corresponds to the task described by C_4.

3.5 Summary

Although we have introduced a formalism for coupling oscillators with a desired connection specification, we have left many questions about this class of systems unanswered. In this final section of the chapter, we list several conjectures that are informed by extensive simulation experience with various connection graphs. These conjectures, for the most part, concern the *form* of the connection graphs we have introduced and attempt to sidestep the analysis of the zeros of H.

First, let us call a reference field R_C associated with the connection graph C **well formed** if it performs the task specified by C. Next, call a graph C **consistent** if, for any cycle $v_1, ..., v_k$ in C we have

$$C_{v_1,v_k} \prod_{j=1}^{k-1} C_{v_j,v_{j+1}} = 1.$$

Thus, a graph is consistent if it does not specify transitively that an oscillator be out of phase with itself. The first conjecture is: 1) If R_C is well formed then C is consistent and connected. We have no hope of proving the converse since the graph C_{bad} in Example 3.4.2 is consistent but $R_{C_{bad}}$ is not well formed. However, we could begin to add edges to C that are consistent with edges already in C. This leads us to the next conjecture: 2) If C is connected then there exists a consistent graph C' such that $C \subseteq C'$ and $R_{C'}$ is well formed.

By examining a large number of consistent, connected graphs it is apparent that cycles often cause R_C to be not well formed. Thus, another conjecture is: 3) If C is a tree then R_C is well formed. Furthermore, trees made up of well formed subgraphs seem to be well formed. That is, suppose C_1, ..., C_n are well formed and C_{tree} is a tree containing exactly one node from each C_i. Put

$$C = C_{tree} \cup \bigcup_{i=1}^{n} C_i.$$

Then the conjecture is: 4) R_C is well formed. A corollary to (3) that is related to (2) is: 3') If C is consistent and T is a spanning tree of C, then R_T is well formed and performs the task specified by C.

The coupling function we use for these graphs is the sine function. The next conjecture is that no other coupling function performs better. Specifically, suppose R'_C is defined by

$$\dot{\phi}_i = 1 - \sum_{j=1}^{n} C_{i,j} f(\phi_i - \phi_j).$$

Then the conjecture is: 5) R'_C with $f(x) = \sin(x)$ is well formed if and only if R'_C with $f(x)$ equal to any other 2π periodic function is well formed.

More generally, we desire a set of graph operations that preserve well formedness. The tree construction above is one such (n-ary) operation. Furthermore, we desire a complete set: The set of all well formed reference fields should be obtained from the closure of some set of basic graphs with respect to a set of well formedness preserving operations on graphs. Such a theory would eliminate the need to do an analysis of the solutions to H, as done in the examples above, and would be an important and prime example of the compositional approach reviewed in Chapter 2 applied to continuous control systems. Investigating the answers to these questions represents, hopefully, future work by the author and his collaborators. The simple reference fields we are able to construct so far, however, do have many uses. These are explored in the next chapters.

CHAPTER 4

Composing Intermittent Contact Systems

To demonstrate the relevance of the reference dynamical systems constructed in the previous chapter to robot control, we show in this chapter a means of constructing controllers for a kind of intermittent control problem where there is at least one degree of freedom that may only be actuated under certain circumstances. For example, we consider the task of bouncing a ball on a paddle (a kind of juggling), wherein the robot may only actuate the ball during collisions. The rest of the time the ball is under the influence of gravity alone. The other example we consider is the control of hopping robots where there is a flight phase. While on the ground, the robot has some affordance over its trajectory. While in the air, it does not. The question we address is whether the assumption of continuous actuation made in Chapter 3 can be relaxed, allowing us to use the cyclic reference fields (3.5) we have defined as the basis of control algorithms for these systems. The answer so far seems to be yes.

Both of these tasks, juggling and hopping, are cyclic. Furthermore, the height of the bouncing ball (or the hopping robot) corresponds to phase velocity: the higher the ball goes, the more energy it has, the longer it takes to complete a cycle. In Section 4.1 we describe how to change from the body coordinates of these systems to phase coordinates which are more natural for dealing with cyclic systems — the first step toward casting the problem as a phase regulation problem. In Section 4.2 we describe the tasks of bouncing a single ball and hopping a single leg. The main difference between the two is that bouncing a ball to a certain height is done with deadbeat[1]control, while hopping to a certain height is achieved only asymptotically.

Then we show how to juggle two balls with one paddle and how to synchronize two hopping robots, using as a basis the reference field (3.4), and analyze the stability of each control algorithm.

.

4.1 Phase Coordinates for Intermittent Systems

We first show how to obtain phase coordinates for intermittent cyclic dynamical systems, by which we mean systems for which a global cross section can be found. Let $f^t : \mathbb{R} \times X \to X$ be a flow on X. Formally, a global cross section Σ is a connected codimension one submanifold of X which transversally intersects every flowline. For any point $x \in X$, define the **time to return** of x to be

$$t^+(x) = min\{t > 0 \mid f^t(x) \in \Sigma\} \tag{4.1}$$

and define the **time since return** of x to be

$$t^-(x) = min\{t \geq 0 \mid f^{-t}(x) \in \Sigma\} \ . \tag{4.2}$$

The *first return map*, $p : \Sigma \to \Sigma$, is the discrete, real valued map given by $p(x) = f^{t^+}(x)$. Let $s(x) = t^+(x) + t^-(x)$. s is the time it takes the system starting at the point $f^{t^-}(x) \in \Sigma$ to reach Σ again. Now, define the **phase** of a point x by

$$\phi(x) = 2\pi \frac{t^-(x)}{s(x)}. \tag{4.3}$$

Notice that the rate of change of phase, $\dot{\phi}$, is equal to $2\pi/s$. The relationship of these functions to Σ is shown in Figure 4.1. Therefore, $\dot{\phi}$ is constant or piecewise constant, changing only when the state passes through Σ.

In the examples we give in Section 4.2, the function $h : X \to Y$, defined by $h(x, \dot{x}) = (\phi, \dot{\phi})$, is actually a change of coordinates where X is the space of positions and velocity pairs and $Y = S^1 \times \mathbb{R}^+$. In juggling, we use the section $\Sigma \subseteq X$ defined

[1]By deadbeat control in this case we mean that from the environment model an appropriate paddle velocity may be determined so that the ball achieves a desired height in one hit. This is not the case in hopping, given the spring stiffening control authority assumed. Because deadbeat control is not very robust to model uncertainty, usually a feedback term is added in similar to that in (4.7)

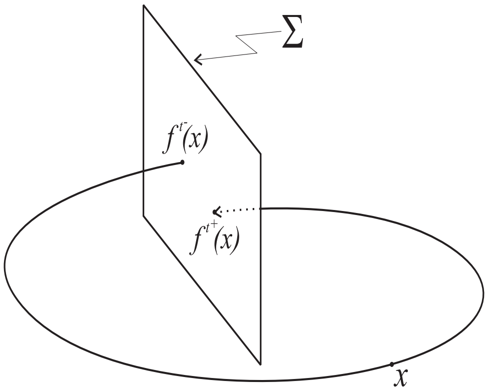


Figure 4.1: The relationship between $t^-(x)$, $t^+(x)$ and x.

by $x = 0$ which corresponds to the set of states where the robot may contact (and thereby actuate) the system. In hopping we use the section in X defined by $\dot{x} = 0$ and $x < 0$ which corresponds to the lowest point in a hop. See Sections 4.2.1 and 4.2.2 for the definitions of these systems. By construction, $h(\Sigma)$ will be given by the set $\mathcal{C} = \{(0, \dot{\phi}) \mid \dot{\phi} \in \mathbb{R}^+\}$ in both examples. In these intermittent control situations, it is only in $\mathcal{C}$ that $\dot{\phi}$ may be altered by the control input u. That is, we change $\dot{\phi}$ according to a control policy u to get the return map in phase coordinates $p' : \mathcal{C} \to \mathcal{C}$ given by $p'(0, \dot{\phi}) = (0, u(\dot{\phi}))$. We design the controller so that there is a unique and stable fixed point at some desired phase velocity $\dot{\phi}^* = \omega$.

Of course we really want to control the system so that the return map p has a stable fixed point at some x^*. Whether or not $h^{-1}(0, \omega) = (0, \dot{x}^*)$ depends on the dimension of Σ. If $dim\ \Sigma = 1$, as it will be in the examples we supply, then the preimage of ω is indeed a point. Otherwise there must be some additional control mechanism which regulates the remaining degrees of freedom.

The main point of this section concerns the composition or interleaving of two such cyclic systems. That is, we suppose we have the system $(x_1, \dot{x}_1, x_2, \dot{x}_2) \in X^2$ with corresponding phase coordinates $(\phi_1, \dot{\phi}_1, \phi_2, \dot{\phi}_2) \in Y^2$. As before, system i may only be actuated when $\phi_i = 0$. In the examples we consider, we suppose that it is undesirable for the systems to be actuated simultaneously. Thus, the set of states where $\phi_1 = \phi_2 = 0$ should be repelling. One task that realizes these goals is (M, G)

where

$$\begin{aligned} M &\triangleq \{(\phi_1, \dot{\phi}_1, \phi_2, \dot{\phi}_2) \mid \phi_1 = \phi_2 + \pi \pmod{2\pi}\} \\ G(\phi_1, \phi_2) &\triangleq (\dot{\phi}_1, \dot{\phi}_2) = (\omega, \omega) \ . \end{aligned} \tag{4.4}$$

The constraint $\phi_1 = \phi_2 + \pi \pmod{2\pi}$ encodes our desire to have the pair of phases as far from the situation $\phi_1 = \phi_2 = 0$ as possible, which in juggling, for example, corresponds to the case where both balls collide with the paddle simultaneously.

To analyze and control such a system, we restrict our attention to the sections $\Sigma_1 \subseteq Y^2$ and $\Sigma_2 \subseteq Y^2$ defined by $\phi_1 = 0$ and $\phi_2 = 0$ respectively. For now, suppose that the flow alternates between the two sections. Let $g^t = H \circ F^t \circ H^{-1}$ be the flow in Y^2 conjugate to the flow in X^2 where $F = (f, f)$ and $H = (h, h)$ and

$$\tau_i(w) = \min\{\tau > 0 \mid g^\tau(w) \in \Sigma_{3-i}\} \ .$$

Start with a point $w \in \Sigma_1$. Let $w' = g^{\tau_1}(w)$ and $w'' = g^{\tau_2}(w')$. We have $w' \in \Sigma_2$ and $w'' \in \Sigma_1$, so we have defined the return map on Σ_1. Now since g is parameterized by the control inputs u_1 and u_2 we get

$$\begin{aligned} w = (0, \dot{\phi}_1, \phi_2, \dot{\phi}_2) &\mapsto \ w' = (\phi_1', u_1, 0, \dot{\phi}_2) \\ &\mapsto \ w'' = (0, u_1, \phi_2', u_2) \ . \end{aligned}$$

Thus, the phase velocity updates $u_1(w)$ and $u_2(w')$ must be found so that (4.4) is achieved. See Figure 4.2(a) for an illustration of the return map. This derivation is based on the assumption that there is direct immediate control over phase velocity during updates, which turns out to be the case in juggling. In hopping, however, there is only asymptotic control over phase velocity updates. This case is shown in the more general Figure 4.2(b) where the delay between the assertion of control and its effect on phase velocity is represented by the function g which maps a control input and current phase velocity to the updated phase velocity. In Section 4.2.2 we describe in detail the form of g which, as mentioned, is nontrivial in the case of hopping.

In Section 4.5, we will show that simply choosing $u_1 = \mathcal{R}_1(\phi_1', 0)$, the first component of a 1:1 reference field as in (3.4), and $u_2 = \mathcal{R}_2(0, \phi_2')$ for well chosen values of κ_1 and κ_2 results in a controlled system that realizes the task (3.3.1). First we

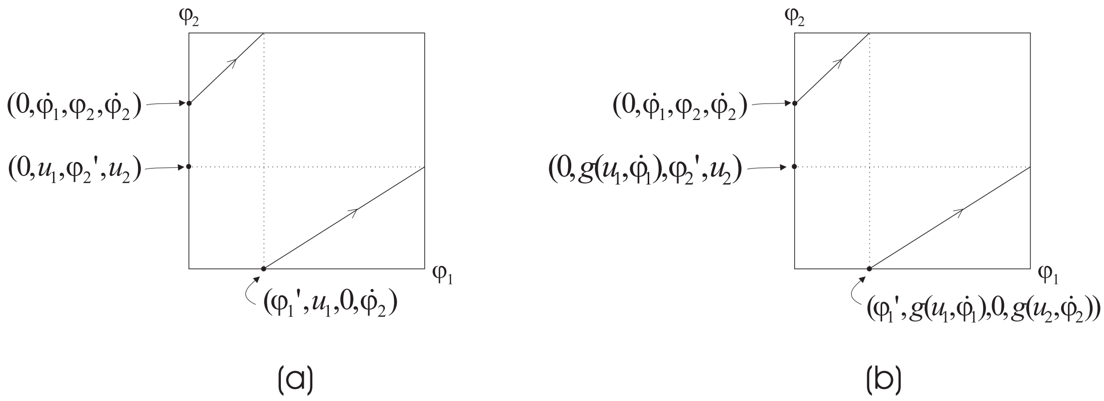


Figure 4.2: Construction of the return map for a two degree of freedom intermittent contact system. (a) Control of individual phase velocities is deadbeat. (b) Control of individual phase velocities is asymptotic.

describe the two systems and their control laws in detail, describing in particular how to obtain the phase coordinates of each.

4.2 Two Examples of Intermittent Contact Tasks

In Section 4.2.1 we consider as an example of a cyclic, intermittent contact problem, the task of bouncing a ball vertically on a piston to a desired height. In this fairly simple task, the paddle can hit the ball at just the right velocity to achieve the desired height in one collision (assuming such actuation is within the torque limits of the paddle), as described in [14]. In Section 4.2.2 we describe a model of a one degree of freedom hopping robot, inspired by Raibert's hopper [67], similar to that studied in [46]. In this model, the robot hops on a spring loaded leg and has control over the stiffness of the spring. We assume that it may instantaneously change the stiffness of the spring just before the decompression phase, at the point of maximal compression, thereby roughly modeling the effect of a pneumatic piston of the type used by Raibert in the design of his hopping robots [67]. With the control algorithm we give for this task, the robot may only *approach* the desired hopping height asymptotically, so that the discussion at the end of Section 4.1 only approximately applies. Nevertheless, when regulating two such hoppers, the same control idea — to "sample" the reference field at the cross section — applies in this situation as well. In Section 4.3 we describe how to juggle two balls at once and how to synchronize to hoppers. Then in Section 4.5 we give analytical and numerical evidence that this method is correct.

4.2.1 Juggling

Consider a system wherein a paddle with position p controls a single ball with position x to bounce, repeatedly, to a prespecified apex. Suppose the paddle always strikes the ball at $p = x = 0$ and instantaneously changes its velocity according to the rule

$$\dot{x}_{new} = -\alpha\dot{x} + (1+\alpha)\dot{p}. \tag{4.5}$$

The constant $0 < \alpha \leq 1$ is the coefficient of restitution. We suppose that the velocity of the paddle is essentially unchanged by collisions. Evidently, a paddle velocity of $\dot{p} = (\alpha - 1)/(\alpha + 1)\dot{x}$ will set $\dot{x}_{new} = -\dot{x}$. Now define $\eta = \frac{1}{2}\dot{x}^2 + \gamma x$ to be the total energy of the ball, where $\gamma \approx 9.81$ describes the force due to gravity. By conservation of energy, $\dot{\eta} = 0$ between collisions. Set η^* to be a desired energy (corresponding to a desired apex). Define a reference trajectory for the paddle to follow by

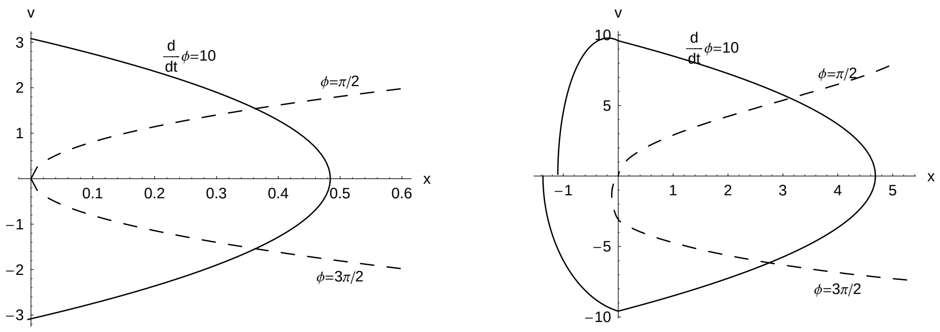
$$\mu = cx \tag{4.6}$$

where

$$c = \frac{\alpha - 1}{\alpha + 1} + k(\eta - \eta^*) \tag{4.7}$$

is constant between collisions. μ is called a *mirror law* because it defines a distorted "mirror" of the ball's trajectory. As the ball goes up the paddle goes down and *vice versa*. The gain, k, adjusts how aggressively the controller minimizes the energy error. If we assume that the paddle follows the reference trajectory very closely, the dynamics of the paddle are then a function of the ball position so that the system is effectively two dimensional (the position and velocity of the ball). It can be shown [14] to drive the ball to the height corresponding to the energy η^*.

Using (4.3) we define the phase ϕ of the ball so that $\phi = 0$ when it leaves the paddle, $\phi = \pi$ at the highest point of its flight, and $\phi = 2\pi$ as it hits the paddle again. Suppose the ball rebounds from a collision with the paddle with velocity



(a) (b)

Figure 4.3: The relationship between phase, phase velocity and the orbits of the (a) juggling and (b) hopping systems. Shown with black lines are typical one-cycle orbits, corresponding to a certain phase velocity, and with dashed lines, the curves $\phi = \pi/2$ and $\phi = 3\pi/2$.

$\dot{x}_0$. By integrating the dynamics $\ddot{x} = -\gamma$ and noting that collisions occur when $x = 0$, we obtain the time since the last impact and the time between impacts — computationally effective instances of (4.1) and (4.2) — as

$$t^- = \frac{\dot{x}_0 - \dot{x}}{\gamma} \quad \text{and} \quad s = t^- + t^+ = \frac{2\dot{x}_0}{\gamma} \tag{4.8}$$

respectively. The change of coordinates $h : (\mathbb{R}^+ \times \mathbb{R}) - (0,0) \to S^1 \times \mathbb{R}^+$ from ball coordinates to phase coordinates is given by $h(x, \dot{x}) = (\phi, \dot{\phi})$ where, following the recipe (4.3), we take

$$\phi = \frac{\pi(\dot{x}_0 - \dot{x})}{\dot{x}_0} \quad \text{and} \quad \dot{\phi} = \frac{\pi\gamma}{\dot{x}_0} . \tag{4.9}$$

In Figure 4.3(a) we illustrate the relationship between phase, phase velocity and the orbits of the juggling model.

4.2.2 Hopping

We model a single, vertical hopping leg, a mass $m = 1$ attached to a massless spring leg, by a dynamical system with three discrete modes: flight, compression and decompression. The latter two modes each have the dynamics of a linear, damped spring. The flight mode is entered again once the leg has reached its full extension.

The equations of motion are

$$\ddot{x} = \begin{cases} -g & \text{if } x > 0 & \text{(flight)} \\ -\omega^2(1+\beta^2)x - 2\omega\beta\dot{x} & \text{if } x < 0 \wedge \dot{x} < 0 & \text{(compression)} \\ -\omega_2^2(1+\beta_2^2)x - 2\omega_2\beta_2\dot{x} & \text{if } x < 0 \wedge \dot{x} > 0 & \text{(decompression)} \end{cases}$$

where ω and β are parameters which determine the spring stiffness $\omega^2(1+\beta^2)$ and damping $2\omega\beta$ during compression. This model is similar to that studied in [46] where a period of thrust at the beginning of decompression was used to stabilize the hopper. We abstract the dynamics of thrust and suppose that, during decompression, thrust simply results in a change in spring stiffness and damping. Thus, ω_2 and β_2 are control inputs in our model.

Let x_b be the lowest point that the robot reaches in a given hop, just before decompression. In Appendix B we show that choosing $\beta_2 = \beta$ and $\omega_2 = \omega\tau(x_b)$ where

$$\tau = (1-k_b)e^{\beta\pi}/(1-x_b)$$

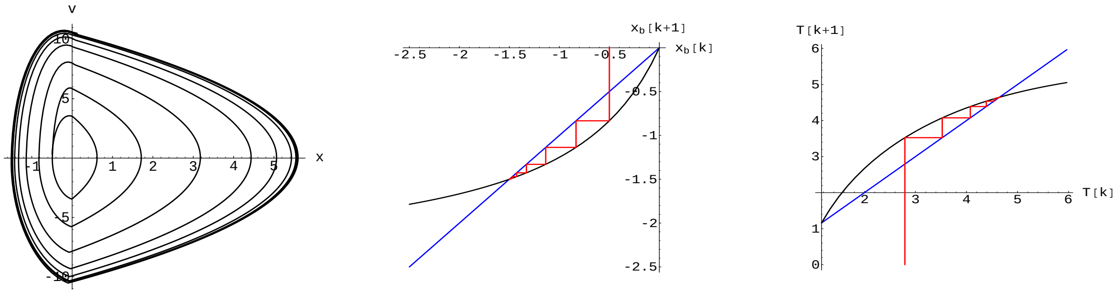
results in a feedback controller that stabilizes the system to have its maximal compression point at k_b. In fact, it can be shown that the discrete, real-valued *return map* $f : \mathbb{R}^- \to \mathbb{R}^-$ that takes the maximal compression point of cycle k to the maximal compression point of cycle $k+1$ is

$$x_{b,next} = f(x_b) = \frac{(1-k_b)x_b}{1-x_b}. \tag{4.10}$$

To determine the phase of this system, it suffices to derive the period, $s(x_b)$, of a cycle starting at $(x_b, 0)$. The value of s is obtained by summing the decompression time t_d, the flight time t_f, and the compression time t_c. It is shown in Appendix B that

$$\begin{aligned} s(x_b) &= t_d + t_f + t_c \\ &= \frac{(\pi - \theta_l)e^{\beta\pi}(1-x_b)}{1-k_b} \\ &\quad -\frac{2}{\gamma}\omega(1-k_b)\sqrt{1+\beta^2}e^{\beta\theta_l}\left(\frac{x_b}{1-x_b}\right) + \frac{\theta_l}{\omega} \end{aligned} \tag{4.11}$$

where $\theta_l \doteq \tan^{-1}(\frac{1}{\beta})$. Given the period corresponding to a particular x_b, we define the phase of a point $(x, \dot{x})$ to be $\phi(x, \dot{x}) = 2\pi t^-(x, \dot{x})/s(x_b)$. In Figure 4.3(b) we illustrate the relationship between phase, phase velocity and the orbits of the juggling model.



(a) (b) (c)

Figure 4.4: (a) A plot of a typical orbit of the system described in (4.10) in position-velocity space. (b) A plot of $f(x_b)$ showing iterations of the return map for the same system. (c) A plot of $g(T)$, the conjugate map, showing corresponding iterations. Note that the convergence to the set-point is asymptotic in this system. For these simulations, $\omega = 6$, $\beta = 0.1$, $k_b = -1.5$.

It can be shown that s is a diffeomorphism on $(-\infty, 0)$. We may, therefore, work equally well with the conjugate map,

$$g(T) \doteq s \circ f \circ s^{-1}(T) \tag{4.12}$$

representing each orbit of the system (4.10) uniquely by it's period. The functions f and g are plotted, along with sample orbits, in Figure 4.4.

4.2.3 Controlling Phase Velocity

By changing η^* in (4.7) we change the desired energy of the ball. This corresponds to changing the desired phase velocity which is $\pi\gamma/\sqrt{2\eta^*}$. Thus, η^*, or more generally the coefficient c in the mirror law (4.6), can be used as a control input to the single juggle. This idea is used in Section 4.3.1 to combine two two-juggles. Similarly, the gain k_b in (4.10) can be used as a control input which affects the point x_b of maximal compression in a hop. Of course, x_b corresponds to the period, as shown in Figure 4.4, which is the same as the phase velocity. The value of k_b is used to change the periods of two hoppers in Section 4.4 so that the synchronize. The responses of (4.10) and (4.12)to changing k_b, for example, is illustrated in a simulation in Figure 4.5.

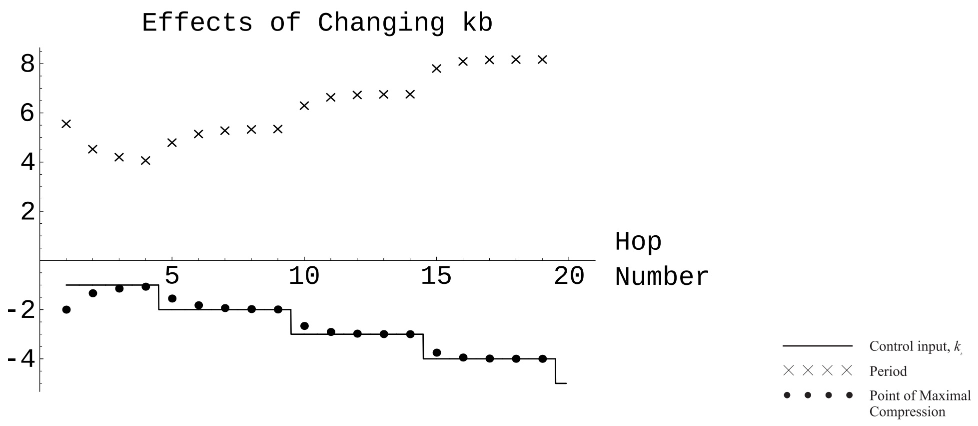


Figure 4.5: The responses of (4.10) and (4.12) to changing k_b

4.3 Implementing the Reference Dynamics

Knowing how to juggle one ball, or control one leg, should somehow lead us to a way of juggling two balls or synchronizing the controllers of two legs. We now show how to use the reference field (3.4) in the case that $A\!:\!B = 1\!:\!1$ to accomplish both of these tasks with only slight modifications to the control algorithms presented above.

4.3.1 Juggling Two Balls

For a two ball system with ball positions x_1 and x_2, we obtain two phases ϕ_1 and ϕ_2. The velocity $\dot{\phi}_i$ is reset instantaneously upon collisions, corresponding to the update rule (4.5).

We next take advantage of the fact that the flow $G^t = H \circ F^t \circ H^{-1}$, described in Section 4.1 and instantiated here, has the very simple form $(y_1, \dot{y}_1, y_2, \dot{y}_2) \mapsto (y_1 + \dot{y}_1 t, \dot{y}_1, y_2 + \dot{y}_2 t, \dot{y}_2)$ between collisions. For each ball i we define a mirror law, μ_i which the paddle should follow when it is about to hit ball i. First, define the lookahead function $C_1(\phi_1, \phi_2)$ to be the phase of ball two when the next ball one collision occurs. Then

$$C_1(\phi_1, \phi_2) = \frac{\dot{\phi}_2}{\dot{\phi}_1}(2\pi - \phi_1) + \phi_2$$

Now, for the first ball we require that, after its next collision,

$$\dot{\phi}_{1,new} = \frac{-\pi\gamma}{\dot{x}_{new}} = \frac{-\pi\gamma}{-\alpha\dot{x}_1 + (1+\alpha)c_1\dot{x}_1} = \mathcal{R}_1 \circ C_1(\phi_1, \phi_2) \tag{4.13}$$

where c_1 is the coefficient in the mirror law trajectory $\mu_1 = c_1 x_1$ and $\mathcal{R}_1(\phi)$ is the first component of the reference field (3.4) evaluated at (ϕ_1, ϕ_2). Let η_1 be the energy of the first ball. Solving for c_1 and using the fact that when $x_1 = 0$, the potential energy is 0 so that $\dot{x}_1 = \sqrt{2\eta_1}$, gives

$$c_1 = \frac{1}{(1+\alpha)\sqrt{2\eta_1}} \left[\alpha\sqrt{2\eta_1} - \frac{\pi\gamma}{\mathcal{R}_1 \circ C_1(\phi_1, \phi_2)} \right]. \tag{4.14}$$

A similar expression for c_2 can be obtained in terms of $\mathcal{R}_2 \circ C_2$. This gives us a mirror law for each ball. Combining these trajectories into a single trajectory of the form

$$\mu = s\mu_1 + (1-s)\mu_2 \tag{4.15}$$

that allows the paddle to manage both systems requires an attention function $s : \mathbb{T}^2 \to [0,1]$ which is 1 before ball one hits and 0 before ball two hits. The construction of s is relegated to Appendix A because it would otherwise be a distraction to the problem at hand. We will assume in Section 4.5 that, away from the situation where both balls strike the paddle simultaneously, the paddle can service both mirror laws in an interleaved fashion.

4.4 Synchronized Hopping

Now suppose we have two physically unconnected hoppers, operating simultaneously, with states $(x_1, \dot{x}_1)$ and $(x_2, \dot{x}_2)$. We will show how to control both hoppers so that they are kept out of phase (one is at its highest point while the other is at its lowest point) and so that they stabilize at a desired hopping height x_b^* (or period T^*). We do this essentially by changing the set-points, now denoted $k_{b,i}$, for each hopper according to the phase of the other hopper. This corresponds to changing the period and thus allows us to regulate the relative phase of the hoppers.

To apply our phase regulation algorithm we reset the gains $k_{b,i}$, each time a leg reaches its lowest point, according to the reference field (3.4) so that

$$k_{b,i} \leftarrow \mathcal{R}(\phi_j) \doteq k_b - k_s \sin(\phi_j) \tag{4.16}$$

where $j = 3 - i$ and k_s is a gain about which we will have more to say later. The parameter k_b sets the desired lowest point in a cycle (which defines the hopping height and, equivalently, the period). Recall that k_b corresponds to period. It appears in the first term of the phase regulation expression instead of phase velocity because it is convenient to later analysis. Using the fact that changing $x_{b,i}$ is equivalent to changing the period T_i, this amounts to a period adjustment scheme for each leg that pushes them out of phase with each other. However, a leg does not respond immediately to the reset because control is asymptotic and not deadbeat. It must, therefore, be shown that this simple method indeed achieves the desired result.

We have defined a system that may be described by the state vector

$$x = (\phi_1, \phi_2, T_1, T_2) \in \mathbb{T} \times \mathbb{R}^+ \times \mathbb{R}^+$$

which evolves as follows. We have $\dot{\phi}_i = 2\pi/T_i$ until some ϕ_i becomes $2\pi \equiv 0$. At this point, its desired hopping height (equivalently, period) is changed according to (4.16) and the period is reset according to the assignment $T_i \leftarrow g_{k_{b,i}}(T_i)$. The system then continues similarly.

4.5 Analysis

We now analyze the local stability of the controlled systems described in Section 4.3. To analyze such systems, we derive the *return map* of each as described in Section 4.1. We let $\Sigma_1 = \{(\phi_1, \phi_2, \dot{\phi}_1, \dot{\phi}_2) \mid \phi_1 = 0\}$ and define a map $F : \Sigma_1 \to \Sigma_1$ that gives the phase of oscillator two, and the phase velocities of both oscillators, just before the first oscillator's phase becomes zero. — under the assumption that zero phase crossings alternate. That is, start with a point $\mathbf{w} \in \Sigma_1$, integrate the system forward to obtain a point in Σ_2, then integrate again to get a point in $F(\mathbf{w}) \in \Sigma_1$. Then, compute the Jacobian $J_{\mathbf{w}^*}$ at the fixed point $\mathbf{w}^*$ of F corresponding to the out of phase situation given by (4.4). If the eigenvalues of the $J_{\mathbf{w}^*}$ lie within the unit circle in the complex plane, then $\mathbf{w}^*$ is a stable fixed point of F. In both models we show this to be the case for certain parameters of the system.

Notice that when $A : B = 1 : 1$, then $\mathcal{R}(0, \phi_2) = \mathcal{R}(\phi_1, 0)$. To simplify notation in this section, we redefine $\mathcal{R} : S^1 \to \mathbb{R}$ to be the reference field restricted to $\phi_1 = 0$. Therefore, with $A\!:\!B = 1\!:\!1$,

$$\mathcal{R}(\phi) = \kappa_1 - \kappa_2 \sin(\phi). \tag{4.17}$$

4.5.1 Local Analysis of Juggling

Supposing that the paddle in the juggling system exactly tracks the reference trajectory (4.15), we may consider the juggling system to be equivalent to the system $(\phi_1, \phi_2, \dot{\phi}_1, \dot{\phi}_2) \in \mathbb{T}^2 \times \mathbb{R}^2$ where $\dot{\phi}_i$ is constant except for discrete jumps made when $\phi_i = 0$. These jumps are governed by the reference field (3.4). That is, when $\phi_i = 0$, $\dot{\phi}_i$ instantaneously becomes $\mathcal{R}(\phi_j)$.

A point in Σ_1 has the form $w = (0, \phi_2, \dot{\phi}_1, \dot{\phi}_2)$. This maps to the point $w' = (C_1, 0, \mathcal{R}(\phi_2), \dot{\phi}_2) \in \Sigma_2$ where C_1 is the phase of the first system when the trajectory of the total system first intersects Σ_2. w' in turn maps to the point $f(w) = (0, C_2, \mathcal{R}(\phi_1), \mathcal{R}(C_1))$ where C_2 is the phase of the second system when the trajectory next intersects Σ_1. The phases C_1 and C_2, which can be obtained via the point-slope formula for a line (in the ϕ_1, ϕ_2 plane), are given by

$$C_1 = \frac{\mathcal{R}(\phi_2)}{\dot{\phi}_2}(2\pi - \phi_2) \ \text{ and } \ C_2 = \frac{\mathcal{R}(C_1)}{\mathcal{R}(\phi_2)}(2\pi - C_1)\ . \tag{4.18}$$

Let $(x, y, z) = (\phi_2, \dot{\phi}_1, \dot{\phi}_2)$. Then, expanding $f(w)$, we obtain a discrete, real valued map on Σ_2 given by

$$\begin{aligned}
x_{k+1} &= \frac{\mathcal{R}\left[\dfrac{\mathcal{R}(x_k)}{z_k}(2\pi - x_k)\right]}{\mathcal{R}(x_k)}\left[2\pi - \frac{\mathcal{R}(x_k)}{z_k}(2\pi - x_k)\right] \\
y_{k+1} &= \mathcal{R}(x_k) \\
z_{k+1} &= \mathcal{R}\left[\frac{\mathcal{R}(x_k)}{z_k}(2\pi - x_k)\right] \ .
\end{aligned} \tag{4.19}$$

Since the x and z advance functions are not functions of y, we can treat y as an output of this system. Thus, analytically, it will suffice to treat (4.19) as an iterated map of the the variables $(x, z) \in S^1 \times \mathbb{R}^+$ given by $F(x_k, z_k) = (x_{k+1}, z_{k+1})$. We have the following fixed point conditions:

Proposition 4.5.1 *$F(x, z) = (x, z)$ if and only if $\mathcal{R}(x) = \mathcal{R}(2\pi - x) = z$.*

We omit the proof, which is straightforward algebra (note that the values of x are always taken modulo 2π since $x = \phi_2 \in S^1$). For the reference field we are using, we have:

Corollary 4.5.1 *If $\mathcal{R}(\phi) = \kappa_1 - \kappa_2 \sin(\phi)$, then the only fixed points of F are (π, κ_1) and $(0, \kappa_1)$.*

We wish to show that the first fixed point, (π, κ_1), is stable, since it corresponds to the situation where the two subsystems are out of phase and at the desired velocity. To do this, we examine the Jacobian:

$$J_{(\pi,\kappa_1)}F = \left(\begin{array}{cc} \dfrac{\kappa_1^2 - 3\kappa_1\kappa 2\pi + \kappa_2^2\pi^2}{\kappa_1^2} & \kappa_2\left(\dfrac{\kappa_2\pi}{\kappa_1} - 1\right) \\ \dfrac{\pi(\kappa_1 - \kappa_2\pi)}{\kappa_1^2} & -\dfrac{\kappa_2\pi}{\kappa_1} \end{array} \right). \tag{4.20}$$

Values of κ_1 and κ_2 which guarantee that the eigenvalues of (4.20) lie within the unit circle are not difficult to find. For example, if the desired phase velocity κ_1 is given, then we can choose κ_2, which adjusts how aggressively the balls are pushed out of phase, to be $\kappa_2 = \frac{1}{2}\kappa_1/\pi$:

Proposition 4.5.2 *If $\kappa_2 = \frac{1}{2}\kappa_1/\pi$ then the eigenvalues of $J_{(\pi,\kappa_1)}F$ are both $1/2$. The point (π, κ_1) is a stable fixed point of F under these conditions.*

The proof is just a calculation: simplify (4.20) using the constraint and compute the eigenvalues. In practice, it is not difficult to find other parameters which make F stable. For a given κ_1, we first choose κ_2 to be quite small and increase it slowly while the controller remains aggressive, yet still stable. It is also simple to show

Proposition 4.5.3 *The eigenvalues of $J_{(0,\kappa_1)}F$ are 0 and $1 + \frac{4\pi\kappa_2}{\kappa_1}$. The point $(0, \kappa_1)$ is an unstable fixed point of F.*

This follows from the fact that κ_1 and κ_2 are both positive. Proposition 4.5.3 shows that the situation in which the two balls collide with the paddle simultaneously is repelling: the system is driven away, locally, from this “obstacle”.

4.5.2 Local Analysis of Synchronized Hopping

In deriving the return map of the juggling system, (4.19), we used the fact that the paddle can strike the ball at just the right velocity to realize the reference field directed updates to phase velocity. In the hopping system, however, adjusting the spring stiffness in decompression does not allow for this. That is, in the juggling system, after a ball one hit, the phase velocity is adjusted according to $\dot{\phi}_1 \leftarrow \mathcal{R}(\phi_2)$. However, in the hopping system, when leg one reaches the bottom-most part of a cycle, the spring stiffness is adjusted so that the period of the hopper is reset according to $T_1 \to g_{k_b}(T_1)$ where g is as in (4.12) and $k_b = \mathcal{R}(\phi_2)$ is obtained from the reference trajectory.

Derivation of the Return Map

Define to be $\Sigma = \{(\phi_1, \phi_2, T_1, T_2) \mid \phi_1 = 0\}$. Assuming that resets of the legs alternate, we construct the return map $F : \Sigma \to \Sigma$. We begin with a point $(0, \phi_2, T_1, T_2)$ just before resetting the period of hopper one. This evolves until a reset of hopper two. If we suppose that C_1 is the phase of hopper one just before hopper two is reset, then, just after the reset we have the point $(C_1, 0, g(\mathcal{R}(\phi_2), T_1), T_2)$, where $g(k_b, T) = g_{k_b}(T)$ is as in (4.12). This point evolves back to Σ so that the state just before hopper one is reset for a second time is $(0, C_2, g(\mathcal{R}(\phi_2), T_1), g(\mathcal{R}(C_1), T_2))$ where C_2 is the phase of the second hopper just before the second reset of the first hopper. Calculating C_1 and C_2 we have

$$
\begin{aligned}
C_1 &= \frac{T_2}{g(\mathcal{R}(\phi_2), T_1)}(2\pi - \phi_2) \\
C_2 &= \frac{g(\mathcal{R}(\phi_2), T_1)}{g(\mathcal{R}(C_1), T_2)}(2\pi - C_1).
\end{aligned}
$$

Letting $x = T_1$, $y = \phi_2$ and $z = T_2$ we obtain the three dimensional, discrete, real-valued return map $F(x, y, z) = (x', y', z')$ corresponding to the oscillating system (4.10) defined by

$$
x' = g(\mathcal{R}(y), x)
$$

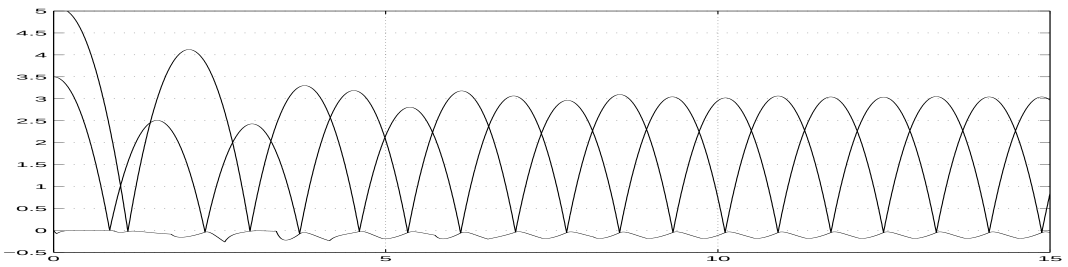


Figure 4.6: A simulation of the 1:1 juggling system described in Section 4.3.1. The positions of the balls and paddle are shown as functions of time.

$$
\begin{aligned}
y' &= \frac{g(\mathcal{R}(y), x)}{g[\mathcal{R}(\frac{z}{g(\mathcal{R}(y),x)}(2\pi - y)), z]} \left[2\pi - \frac{z}{g(\mathcal{R}(y), x)}(2\pi - y)\right] \\
z' &= g\left[R(\frac{z}{g(\mathcal{R}(y), x)}(2\pi - y)), z\right]. \qquad (4.21)
\end{aligned}
$$

It is instructive to compare these equations with the return map (4.19) for juggling — the difference being the appearance of g which accounts for the lag between the assertion of control and its effect.

Local Stability of the Return Map

It can be shown that the point (T^*, π, T^*) is a fixed point of this system, where $T^* = s_{k_b}(k_b)$ is the period corresponding to the set-point k_b. We now have

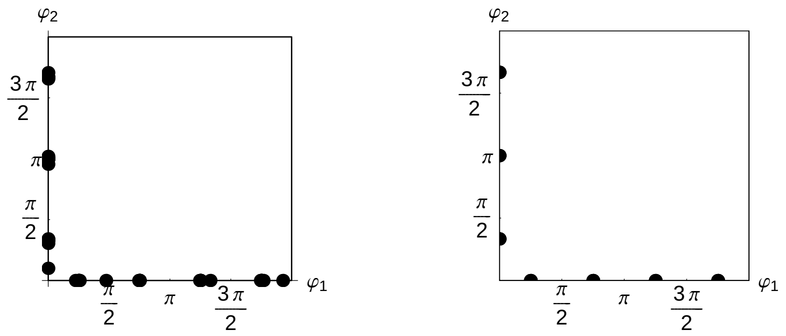
Proposition 4.5.4 *The point (T^*, π, T^*) is a stable fixed point of the system defined by (4.21) when the synchronization gain k_s is chosen to be*

$$\frac{1}{b\pi k_b}(a + c - bk_b)\left[2k_b - 2 + \sqrt{1 - 4k_b + 3k_b^2}\right]. \qquad (4.22)$$

The proof is given in Appendix C

4.6 Numerical Studies and Simulations

We have simulated various combinations of hoppers and jugglers with various couplings and observed that our method stabilizes each system as expected. Figure



(a) (b)

Figure 4.7: A simulation of the juggling system for $A\!:\!B = 3\!:\!4$. Each dot corresponds to a ball-paddle collision in a simulation with 40 collisions. (a) The first 20 collisions are scattered around the equilibrium orbit. The second 20 are at the equilibrium orbit, showing 3 ball one hits for every 4 ball two hits.

4.6 shows a simulation of the $1\!:\!1$ juggling system described in Section 4.3.1. Besides the relationship on the the synchronization gain κ_2 required to show Proposition 4.5.2, other settings of the gains are satisfactory as well. Increasing κ_2 increases the response time of the system. However, a higher setting results in equation (4.19) becoming period two stable, the period four and so on. Eventually, too high a setting gives what appears to be chaotic behavior.

Although Proposition 4.5.2 requires a $1:1$ coupling, we have in fact observed in simulation that any ratio $A\!:\!B$ that we tried could be stabilized provided that κ_2 is small enough. Figure 4.7, for example, shows the hit-points in (the phase of ball two for a ball one collision and *vice verse*) in a simulation of a system with $A\!:\!B = 3\!:\!4$ which stabilizes after only a few ball/paddle collisions.

In Proposition 4.5.4, constraining the value of k_s to a function of k_b achieves analytical simplicity but is hardly necessary. Numerical simulations of the synchronized hopper system suggest a wide interval of k_s settings around the guaranteed values in (4.22) yield stability. In Figure 4.8, we show a simulation starting from arbitrarily chosen initial conditions which eventually stabilizes at the desired hopping height and phase relationship. In our simulations, with k_s suitably small, we could not find initial conditions that did not eventually stabilize — leading us to believe that the

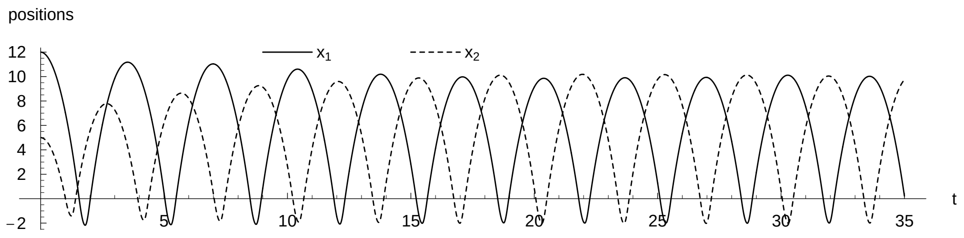


Figure 4.8: A simulation of the 1:1 synchronized hopping system described in Section 4.4. The positions of the leg masses are shown as functions of time.

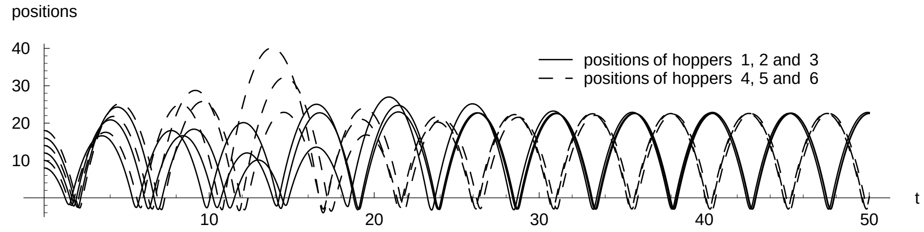
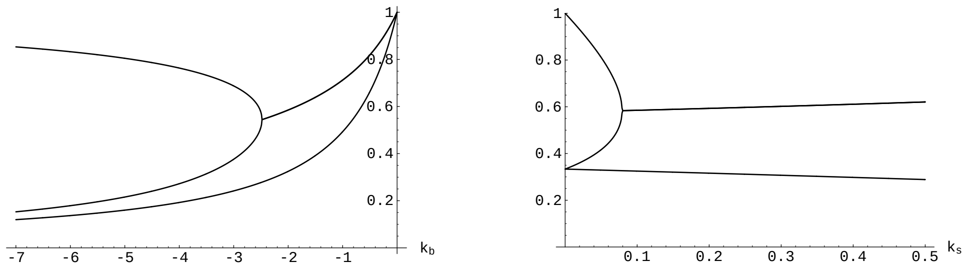


Figure 4.9: A simulation of six hopping robots synchronized into two out of phase groups. The two groups consisting of robots one, two and three and robots four, five and six respectively, are forced to be in phase with other members of their groups.

system is in fact globally asymptotically stable.

Of course, the general theory laid out in Chapter 3 and particularly in 3.3 may be applied to intermittent contact systems with more than two oscillators. Figure 4.9 shows a six-hopper system as an example of this. The connection matrix for the system is (3.1) and thus specifies an *alternating tripod* wherein two groups of three are synchronized out of phase.

We also investigated the eigenvalues of (C.1) numerically without the simplification (4.22). For example, fixing k_s and changing k_b results in the following situation: when $k_b = 0$, the system is not stable (the eigenvalues have magnitude one) because the hopping height is zero and, therefore, the decompression phase never ends. As k_b decreases (resulting in a larger hopping height), the sizes of the eigenvalues decrease for a time and then one of them increases toward one as k_b approaches $-\infty$. If we instead fix k_b and vary k_s, the we observe the following: when $k_s = 0$, there is no coupling and the system is only neutrally stable at the fixed point. As k_s increases, the



(a) (b)

Figure 4.10: (a) Magnitudes of the eigenvalues of (C.1) (without the simplification (4.22)) versus k_b with $\omega = 6$, $\beta = 0.1$ and $k_s = -0.1$. (b) Magnitudes of the eigenvalues versus k_s with $\omega = 6$, $\beta = 0.1$ and $k_b = -2$.

system stabilizes until a certain point, after which the magnitude of one eigenvalue exceeds one. In our simulations, values of k_s larger than the point at which two of the eigenvalues become imaginary resulted in significant overshoot of the fixed point and longer convergence time. We usually chose k_s to be such that the eigenvalues are all real, and this improved performance.

4.7 Discussion

We have described two intermittent contact systems: ball batting and hopping and shown how to use the reference dynamics of Chapter 3 to combine such systems into synchronized version. We believe that these systems are representative of a a larger class of intermittent contact systems in general. For example, it has been shown [25] that the *Stick Insect* regulates the phase relationships between its legs using only the posterior extremal position (PEP) of each leg. The PEP is the back most angle that a leg reaches during a given power stroke, or stance phase. The further back PEP is, the longer the period of a leg's cycle is. If PEP is less far back, the period is shorter. Cruse [25] has proposed and experimentally verified a model of gait regulation for the stick insect which is based on certain communication mechanisms between the legs. Thus, the control of gait in this animal seems to be decentralized *and* intermittent. In [16], Callvitt and Beer analyze the stability of a two legged model of the stick insect with one way communication (from one leg to the other). In [43], we show for our own dynamical model of the stick insect that the reference field idea (where PEP is modulated in a way similar to (4.16)) can mimic a two way version of the Cruse mechanism for synchronization. Our method, furthermore, can be adapted produce to other behaviors by altering the energy function V (as in equation (3.4)). We believe that our method is somewhat more parsimonious and better applicable to the task of controlling robots — as in Chapter 5.

There are several avenues yet to explore regarding the work presented in this chapter. First of all, the analyses in Section 4.5 are local. Although numerically, in simulation these systems seem to have global stability properties, the analysis of two and three dimensional return maps is difficult and has not been done here. Furthermore, we assumed that hit points alternate when constructing the return maps. The transients encountered in other mode sequences need careful attention as well. This has not been done here, although once again, numerically evidence suggests stability. Finally, although it is easy to construct and simulate systems with more than two components being synchronized, these systems are high dimensional indeed. A more sophisticated approach is needed before the stability of such systems can be regularly investigated. Finally, it is possible to compose composed systems hierarchically in the following sense: one a system is phase regulated, its phase can be defined as well. For example, a synchronized pair of hoppers represents a cyclic system itself and can be synchronized with respect to another pair, and so on. Simulations not reported here suggest that the gain to use as an input to a pair of systems is the nominal phase velocity represented by κ_1 in (3.5).

CHAPTER 5

Application to the RHex Robot

The oscillator networks we constructed in Chapter 3, and specifically those similar to the examples in Section 3.4, can be used to produce gait controllers for legged robots. We first saw this in the stick insect model in Chapter 4. In this chapter we consider gait controllers for the six legged RHex robot [74] shown in Figure 5.1(a). RHex is a result of an approach to building robots called functional biomimesis [45], where the design of mechanisms is inspired by biological example in function but not necessarily in detail. The philosophy of functional biomimesis is that practicable engineering designs are derived from biological understanding by developing analogies at the appropriate level of abstraction. Thus, each leg of RHex is a one degree of freedom, spring leg instead of a six degree of freedom leg as might be found on a more appearance-based biomimetic robot. Despite its distinctly unnatural appearance and apparent kinematic simplicity, RHex is designed to mimic certain functions exhibited by sprawled posture, many-legged runners such as the cockroach species Blaberus discoidalis.

Each of RHex's legs is controlled by a single motor, mounted at the hip, with encoders used to measure their angular positions. Although RHex has some sensors (accelerometers and gyroscopes) we consider only the position of each leg in our algorithms — particularly the error between a leg's position and some reference signal. In Section 5.1 we review a very simple (yet successful) centralized, feedforward controller (described in [74]), for generating an alternating tripod gait for the robot, which drives each leg to follow an appropriate reference trajectory. In Section 5.2

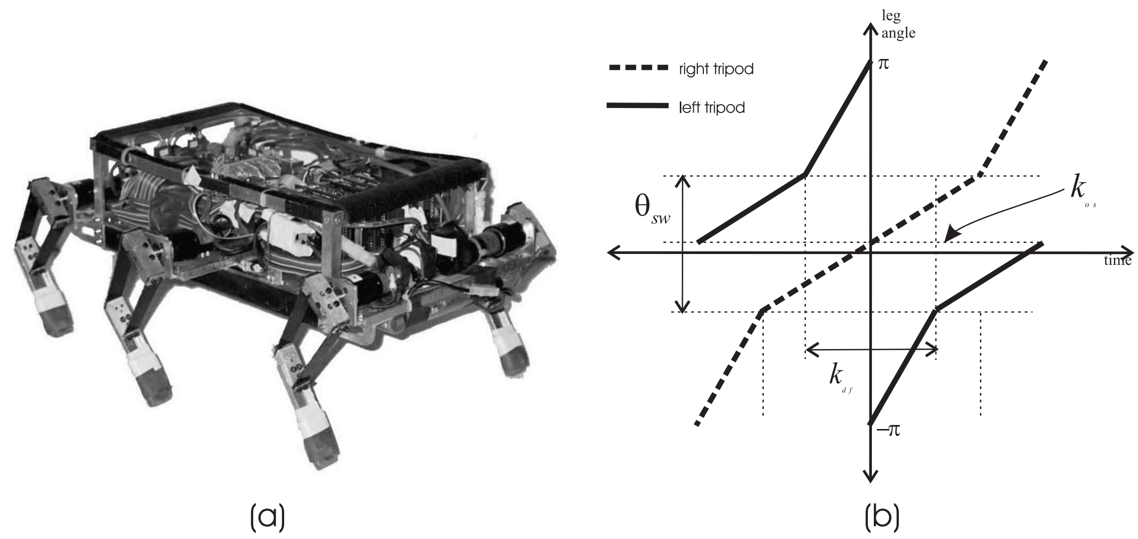


Figure 5.1: (a) The RHex robot has six one DOF, compliant legs. RHex weighs about 7 kg and is roughly the size of a shoe box, about 50 cm × 30 cm. (b) The gaits discussed in this chapter send a fast-slow-fast profile to the legs, as shown here.

we describe an alternative means of controlling RHex, which uses a coupled oscillator network and velocity-based PD control. We show that by tuning a certain gain we can mimic the feedforward controller or investigate the effect of feedback from the legs. In Section 5.3 we report the results of initial experiments with the coupled oscillator controller that suggest that a certain amount of feedback of the sort facilitated by our controller reduces the power consumption of the robot slightly.

5.1 Feedforward Control

The basic part of the centralized, feedforward controller [74] for RHex is a central pattern generator, or CPG, which is in part inspired by the idea of biological pattern generators — units of neural tissue that oscillate in isolation and excite motor activity in intact animals [60, 64]. Beginning with the work of [21] and continuing through more elaborated refinements [22, 52, 47, 28], a flourishing applied mathematical literature has employed dynamical systems theory to model the nature of such feedforward motor control signals, as mentioned in Chapter 1. The feedforward controller for RHex represents a very simple application of the CPG idea.

In RHex, the CPG is a simple, internal oscillator $\phi_{cpg} \in [0, 2\pi]/\{0 \sim 2\pi\}$ with the dynamics

$$\dot{\phi} = v_{cpg}$$

where v_{cpg} is a constant parameter that ultimately affects the speed of walking. Each leg follows a reference trajectory that is a function, f_{prof} illustrated in Figure 5.1(b), of ϕ_{cpg}. More correctly, each leg has its own copy of f_{prof} which it may instantiate with different values of the parameters. The velocity parameter v_{cpg} together with the parameters θ_{sw}, k_{df} and k_{os} in f_{prof}, which correspond to the *sweep angle*, the *duty factor* and the *leg offset* (as illustrated in Figure 5.1(b)) are used to tune the walking behavior.

Denote by $\theta_1, ..., \theta_6$ the angles of the legs. In the most simple mode of the feedforward controller, using PD control, the legs in the first tripod, θ_1, θ_3 and θ_5, are made to track $f_{prof}(\phi_{cpg})$ while legs in the second tripod, θ_2, θ_4 and θ_6, are made to track $f_{prof}(\phi_{cpg} + \pi)$. The result is that the two tripods alternate: While tripod one is in the slow part of the profile, where the legs are likely to be touching the ground, the other tripod is in the fast mode of the profile, swinging over the top of the robot. The duty factor k_{df} controls the amount of overlap the two tripods have in slow (or fast) mode with $k_{df} = 0.5$ corresponding to no overlap. Other modes of operation can be obtained by changing the parameters of the different leg reference trajectory functions. For example, the robot can be made to turn in place by having the legs on one side of the robot track the reverse of their reference trajectories. Also, the robot can be made to turn while walking by increasing k_{os} for legs on one side of the robot and decreasing it for legs on the other side.

With this simple control scheme, RHex is uniquely successful in its ability to "scamper" over rough and variable terrain. Because of its inherent mechanical stability, balance and foot placement are of minimal concern and the legs are used more to propel the body forward. We suspect, however, that more efficient means of locomotion may be possible, particularly if a control scheme which incorporated feedback from the legs (with respect to ground collisions, for example) could be devised. To that end, we introduce a more decentralized controller that allows for feedback, as described in the next section.

5.2 Decentralized Feedback Control

The above controller is centralized in the sense that the CPG sends the same signal (possibly out of phase) to each leg. A decentralized version of the same idea can also be constructed. We start with six *peripheral pattern generators* (PPGs), one for each leg. Each leg tries to match its phase velocity with the PPG. A PPG, like a CPG, is an oscillator. Unlike a CPG, however, the phase velocity of a PPG is not constant; it may be affected by other PPGs or by sensory-motor activity. Each PPG tries to match its phase and phase velocity with the phase of a corresponding leg and also tries to synchronize, with respect to some coordination specification, with the other PPGs. Figure 5.2 illustrates the connection structure of this decentralized scheme. The structure of the internal PPG network can be anything consistent with the fully connected graph shown in gray in the figure where each tripod is out of phase with the other. In practice, we have experimented with the fully connected graph and with the first graph in Section 3.4.

We suppose that we have velocity control of the legs, even though the steady state velocity of a motor for a given voltage command is just that and not the velocity immediately obtained upon application of a particular terminal voltage. Thus, given the present velocity v_i and desired velocity $v_{i,ref}$ of motor i, we apply the voltage

$$V_{command} = m[v_i - k_D(v_i - v_{i,ref})]$$

where m maps steady state velocity to voltage using the parameters of the motor and the attached gear box. k_D is a tunable gain.

The problem of controlling each leg, then, is to provide it with the reference velocity $v_{i,ref}$. Let $x_i \in [0, 2\pi]$ be the position of leg i, let ϕ_i be the phase of PPG i. Then we desire

$$v_{i,ref} = (k_B \sin[x_i - f_{prof}(\phi_i)] + 1)\,\dot{\phi}_i f'_{prof}(\phi_i). \tag{5.1}$$

Thus, if $x_i - f_{prof}(\phi_i) = 0$ — that is, if the position of the leg corresponds to the desired position given by the PPG — then this equation is simply $v_{i,ref} = \dot{\phi}_i f'_{prof}(\phi_i)$,

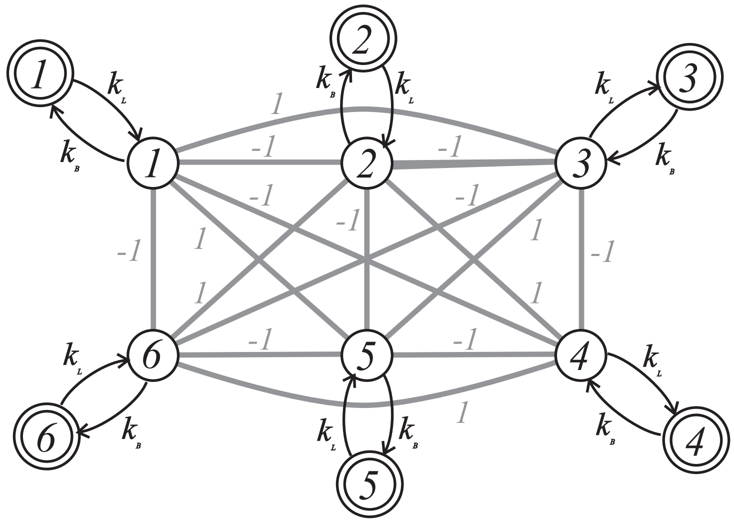


Figure 5.2: The structure of the oscillator network implementing the distributed gait controller for the RHex robot. The connection structure of the internal oscillators is gray, indicating that various subsets of the complete graph illustrated can be used — as long as they perform the task specified (see Chapter 3). The legs are coupled to corresponding oscillators via gains k_L, which determines the strength of the feedback signal from the legs, and k_B, which determines the strength of the feedforward signal from the oscillator.

which is the desired velocity given by the PPG. The sin term is the feedback term and is based (essentially) on the phase difference between the leg and the PPG. The constant gain k_B is used to tune the amount of feedback.

The internal oscillators are connected via a connection matrix (see Chapter 3) C which specifies an alternating tripod that the corresponding oscillator system (in the absence of the legs) can perform. The equation for an individual oscillator is similar to (3.4) with a term added to synchronize the PPG with its leg:

$$\dot{\phi}_i = v_{i,ref} - k_S \sum_{j=1}^{6} C_{i,j} \sin(\phi_i - \phi_j) - k_L \sin(\phi_i - \theta_i) \tag{5.2}$$

where θ_i is the "phase" of leg i and is given by

$$\theta_i = f_{prof}^{-1}(x_i).$$

The gain k_S tunes the internal synchronization strength, and the gain k_L controls the amount of feedback from the leg. When the PPG is perfectly synchronized with the other PPGs and with its leg, its phase velocity is simply v_{cpg}.

In [43] we argue, with respect to a similar control scheme for a very simple model of a planar robot, that the gains k_L and k_S correspond to the various extremes of feedback *versus* feedforward and centralized *versus* decentralized. Thus, if $k_L = 0$ then the system is feedforward and centralized, — essentially the same as the control scheme in Section 5.1, because in the absence of disturbances from the legs, the internal oscillators will synchronize and essentially send the same signal to all the of legs as would a single CPG. If k_L is small, then the gains k_S and k_B seem to tune the degree of decentralization. If they are small, then the PPGs and the legs are more independent of each other. If they are larger, then the PPGs are more yoked to each other, and the legs are less independent.

5.3 Initial Experimental Results

Our initial experiments with the coupled oscillator controller for RHex focus on the feedback/feedforward axis of control and its effect on power consumption. In its present configuration, RHex can operate on level ground for about 30 minutes before its batteries are drained. A more power-efficient gait may increase RHex's running time. One way to gauge the power efficiency of a locomoting robot is with *specific resistance* [30], which for our purposes is defined by

$$s \triangleq \frac{P_{avg}}{mgv} \tag{5.3}$$

where P_{avg} is the average (electrical) power consumption of the robot, m is its mass, g is the force due to gravity and v is the velocity of the robot.

Presently, the electronics and motor driver circuitry of the robot account for about 43.6W. While walking, power consumption rises from this baseline by about 10W, with spikes of up to another 50W when the legs strike the ground or switch from fast to slow. The specific resistance measure takes into account both the power needed to walk and the electronics power. We suspect that future versions of the robot will be more efficient in terms of the power consumption of the electronics so that even

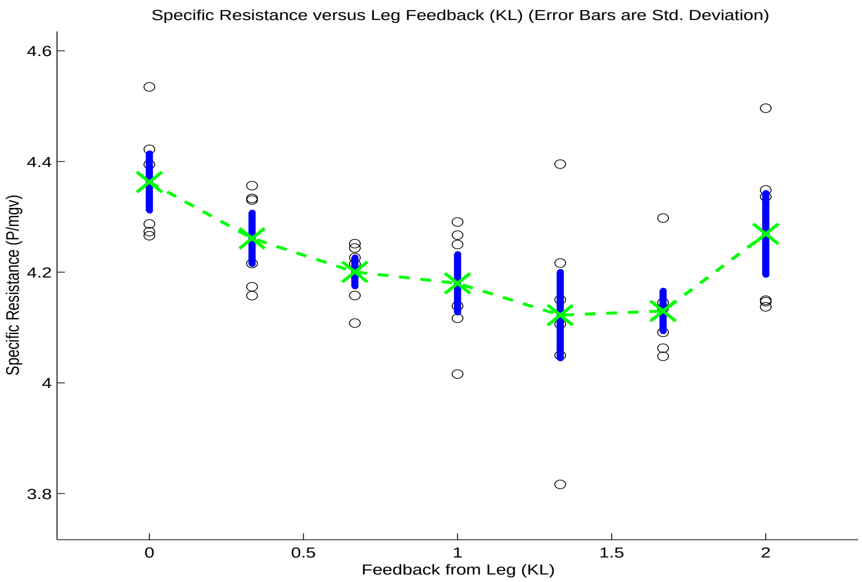


Figure 5.3: Specific resistance, $\frac{P_{avg}}{mgv}$, as k_L varies where P_{avg} is the average electrical power of the robot, m is the mass of the robot, g is the force due to gravity and v is the velocity of the (center of mass of the) robot. For these experiments, $k_B = 1.5$, $k_S = 2.0$, $k_D = 2.25$, $v_{cpg} = 8.5$ seconds, $k_{df} = 0.69$, $\theta_{sw} = 0.61$ radians and $k_d f = 0.69$.

a small drop (as percentage of the total value) in specific resistance for the present version may represent a greater percentage of the same measure in a later version of the robot.

To measure s for RHex, we placed strips of tape on a smooth, level, cement floor parallel and 3m apart. We started RHex walking with the coupled oscillator controller 1m behind the first strip going toward the second strip. When the robot crossed the "starting line," we started, via remote control, logging the time, battery current and voltage as well as the leg positions and phases of the internal oscillators. We stopped logging when the robot crossed the "finish line." From the logged current and voltage data we obtain electrical power as $P = IV$. Average power is computed to be the sum of all power readings divided by the total number of readings. The velocity is computed as $\Delta x/\Delta t = 3\text{m}/(t_1 - t_0)$ where t_0 is the time logging began and t_1 is the time logging ended.

Figure 5.3 shows the effects on specific resistance of varying k_L from 0 (purely

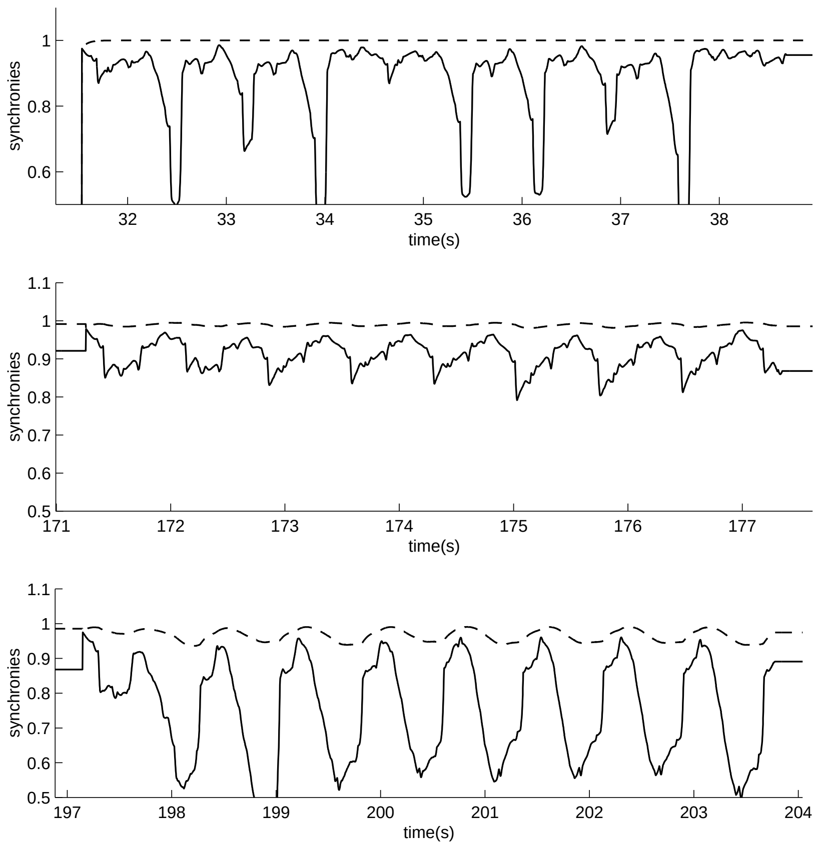


Figure 5.4: Synchrony of the legs (solid line) and synchrony of the internal oscillators (dashed line) versus time for $k_L = \{0, 1, 5\}$ and the rest of the parameters as in Figure 5.3.

feedforward) to 2.0. In all, seven values of k_L were examined. For each value, six experiments were done, represented by the data points. The error bars represent the experimental standard deviation over the six experiments. A small dip in the value of s near $k_L = 1.5$ is apparent in this figure. Evidently, the feedback to the PPG network delivered by the legs has a good effect on power consumption. Beyond 1.5 the measured value of s increases with k_L, and after a certain point ($k_L > 8.0$) the walking controller destabilizes (not shown).

Some intuition about why an increase k_L should lower s can be found by examining the *degree of synchrony* of the internal oscillators and of the legs. For an alternating tripod gait, with legs 1, 3, and 5 in one tripod and legs 2, 4 and 6 in the other, a measure of synchrony is

$$\Phi = \left\| \sum_{j=0}^{6} e^{i(\phi_j + \pi j)} \right\| \tag{5.4}$$

where $i = \sqrt{-1}$. A similar definition of the synchrony of the leg phases is obtained using the θ_is and is denoted by Θ. A value of 0 for these measures means perfectly un-synchronized while a value of 1 means perfectly synchronized. The question is, for various values of k_L, how well do Φ and Θ coincide? Roughly speaking, the idea is that the internal oscillator network is acting like a *model* of the external legs. If the model is good, then Θ and Φ should be close. In Figure 5.4, Θ and Φ are plotted as functions of time for three different values of k_L. In the first plot, $k_L = 0$ so that the internal oscillators synchronize and stay synchronized. Meanwhile, the legs attempt to follow the feedforward signal generated by their corresponding PPGs. When k_L is increased, Φ fluctuates as the legs hit the ground and temporarily desynchronize with their PPGs. The result is that the legs do not have to "do as much work" to synchronize with their PPGs because the PPG phases are closer to the leg phases than they would be without the extra feedback term in (5.2). When k_L is even larger, the internal oscillators are much less synchronized so that the totality of the reference signals they are sending to the legs do not describe an effective alternating tripod gait, and both Θ and Φ fluctuate considerably.

5.4 Further Work on RHex

As these initial experiments show, the parameters in the coupled oscillator controller warrant further investigation. The gain k_L in a sense tunes the strength of the feedback from the legs, defining a "feedback/feedforward axis." The gains k_S and k_B tune the strength of the coupling between oscillators, in a sense defining a "centralized/decentralized axis." Thus, the controller gives us a two axis *design space* to explore for the right combination of gains for optimal power consumption. We have only explored a small part of the space. Ultimately, however, the coupled oscillator controller as described above does not "know" enough about the task of walking. A more informed controller for each leg, one that will not lift off until the other tripod is about to touchdown or one that synchronizes itself with the compression and decompression dynamics of the spring legs is needed. We believe that the coupling mechanisms introduced in Chapter 3 will, nevertheless, apply to synchronizing such controllers.

CHAPTER 6

Conclusion

This thesis represents a subset of our attempts to make formal a simple idea that has already been used more or less intuitively in the lab as matter of common practice. The idea is to compose already working controllers into larger, coupled controllers. The point is to reuse knowledge about how to perform subtasks in order to understand how to perform some global task. The work of Koditschek and Burridge on sequential composition of juggling behaviors [15] and the work of Bühler, Rizzi and Koditschek on juggling two balls with a paddle [14, 12, 73] was the starting point for this investigation, but we have also been informed by other work in computer science and robotics as outlined in Chapter 2. Presently, devising a broadly applicable class of controller compositions, and beginning to propose a general theory, is a very difficult, although fundamentally important, problem. Such a theory could be applied toward controlling or understanding anything composed of many similar parts: agile manufacturing systems where factories need to be built from off-the-shelf components and programmed in time to get a critical product to market; legged locomotion of robots where many legs do essentially the same thing; biological and neuro-mechanical systems, which seem to be decentralized and are certainly composed of similar parts; embedded systems with hundreds of sensing, actuating and communicating processors; and so on. An understanding of controller composition may even address the *symbol to signal* problem, which is the problem of grounding the *meaning* of a symbol in a physical system. We might call a controller a symbol, or try to put differential topology on the same footing as the logical topologies encountered in Domain Theory and seemingly demanded by discrete event supervisory controllers or specification languages. Although we have only begun to understand composition and have only scratched the surface of what is possible, the compositions described in this thesis nevertheless comprise a significant step toward a theory of controller composition.

In developing our compositional approach, we discovered obstacles that many others have undoubtedly encountered before us. Essentially the problem is this: Controllers do not compose transparently unless they are practically decoupled to begin with. Otherwise composition requires no small amount of either luck or perhaps wizardly intuition. We envision a specification language that can describe how a large system is composed of subsystems and how each of the subsystems further decompose. That the actual implementation will have the same discrete structure as the specification may be too much to hope for, however, unless we restrict ourselves to very simple, structured situations. This is exactly what we have done in this thesis and in other work ([39, 42], outlined in Section 2.2). In juggling, the task of the paddle, besides managing the energy of the two balls, is to keep the balls separated in space (as in the three degree of freedom case [73]) and in phase, thereby essentially decoupling them. In the case of the two hoppers are already decoupled. With the RHex controller, we imagine that the ground is only a minor disturbance (which of course it is not) to an otherwise mechanically decoupled set of six legs. In the manufacturing work, concurrently operating controllers are separated into disjoint workspaces. Sequences (or threads) of controllers are obviously decoupled in that they operate at different times. The composition of nonlinear PD horizontal stabilization with ball energy stabilization in the work on ball batting with a paddle [14] or the parallel stabilization of body pitch, hopping height and forward velocity in hopping [67] is more interesting. An argument can be made, however, that these separate tasks operate on different degrees of freedom (or different controllable submanifolds of the state space) that are all but decoupled (and completely decoupled at equilibrium). Robust feedback control takes care of any coupling that might sneak in. Perhaps, then, the task of composing controllers is that of determining how to arrange or manage them so that they *feel* decoupled, either because they are in fact decoupled in time or space or because of the presence of some coupling mechanism, such as the coupled oscillator framework in Chapter 3.

Another theme that has emerged from our investigation of compositional methods is *decentralization*, as indicated by the title of this thesis. As noted in the introduction, models or frameworks for decentralization of control systems have not been developed or are not widely accepted or practicable. Certainly in discrete systems, concurrency models such as Petri Nets are quite useful. But two systems of differential equations look essentially the same even if one is derived from a network and the other is not. Nevertheless, systems whose components are of the form (1.1) do stand out as a new and interesting class of problems — especially if the neighbor relationship changes over time or is subject to interference. The coupled oscillator networks and the Threaded Petri Nets are the first examples we have produced of such decentralized systems. We hope that our methods will apply to other systems and tasks as well.

We next review some unanswered questions raised by this work which have not already been asked in the summaries of the individual chapters.

Reference Fields from Connection Graphs These systems are a wonderful source of complexity. A graph may yield a system that performs the desired task or performs some other task. Its fixed points may or may not be hyperbolic, and in general a complicated repellor/attractor structure arises for which we do not have a general analytical technique. We do not even have a method for solving $H(\mathbf{z}) = \mathbf{0}$. Ultimately what is desired is a computationally effective method for determining the behavior of a system given its graph. Such a method would be quite interesting if it did not rely directly on obtaining and analyzing the zeros of H. It would be more informative and unique if the method instead was based only on the structure of the graph. Suppose for example that predicates such as "C contains such and such a subgraph," "C can be composed from some of these simpler graphs" or "C is (represented by) a string in the language generated by a particular grammar" could be used to determine the behavior of the corresponding system. This would tie together topological dynamics and graph theory in a completely new way. Another way of putting together reference systems is hierarchically. The phase of a reference system can be defined as the phase of its limiting behavior. Thus, two such systems can be phase regulated. This certainly deserves some attention as it captures the idea of composition very nicely.

Using Reference Fields We have used reference fields to control intermittent contact systems and to drive a PD-like velocity controller for RHex. An obvious question remains: How can reference fields be used for systems with other control authorities? What about $\ddot{\phi}_i = u_i$ for example? More interestingly, can phase regulation be combined with the Petri Net cycles discussed in Section 2.2? These systems have *wait modes* which seem on the surface to be antithetical to phase regulation. But perhaps not. One place where one can imagine such a thing working is in a new controller for RHex. The coupled oscillator controller does not "know" about configuration space "obstacles" such as the undesirable situation in which all the legs are up in the air. Wait modes can avoid this and other situations. However, away from such obstacles, phase regulation based on reference fields seems like the right approach. One can imagine situations on an assembly line where such a technique might be useful as well. Finally, the reference field idea itself can be extended to systems other than cyclic dynamical systems and used as a template for producing behaviors for systems with control authorities more complex than $\dot{x} = u$. This idea is similar to Ostrowski's work on nonholonomic systems [62].

Gait Regulation As discussed in the introduction, there is evidence that insects and other animals use networks for coupled oscillators to produce rhythmic movements in their limbs so as to locomote. This is essentially what we have done with our RHex controller. It remains unclear, however, why one would choose a particular connection structure over another and what the role of feedback is in these systems. In Chapter 5 we supplied one simplistic idea of how feedback could be incorporated into a network of oscillators, but certainly there must be more informed ways of doing this. Another aspect of the RHex controller that we explored but did not report on in this thesis (although see [43]) is the *degree of centralization* of the controller, which can be, to a first approximation, related to the gain κ_2 in (3.5) which clearly controls how tightly coupled the oscillators are. A smaller value for κ_2 loosely corresponds to more autonomy for each oscillator. In general, we can (just barely) perceive a two axis design space for controllers like the RHex controller. One axis is feedforward/feedback and the other is centralized/decentralized.[1] Locating a particular point in this design space as being more or less optimal according to some measure (such as specific resistance) is the challenge.

6.1 Final Thoughts

The methods for the decentralized phase regulation of cyclic systems presented in this thesis, and compositional methods in general, are attempts to address the problem of scalability in robotics and control. The approach is inspired by the success of computer engineering, which has taken advantage of the recursive explosion enabled by computer languages to produce complex software systems and VLSI hardware that would not be possible without modularity and abstraction, the hallmarks of a compositional approach. Can robotics and control achieve the same heights? The applications of the future — embedded systems, micromechanical manipulation and assembly, nanosystems, computational biology and chemistry — demand methods that can conquer the complexity that the presence of thousands or millions of interacting components entails. Decentralization, modularity, abstraction and coupling (or communication) mechanisms all seem to be a part of the solution, as demonstrated in this thesis.

[1] This is probably the wrong terminology. Decentralized has to do with where decisions are made and actions are executed. The gain κ_2 really controls how much a particular oscillator conforms to its neighbors.

APPENDICES

APPENDIX A

Construction of Attention Functions

In this appendix we will generally work in the covering space $\mathbb{R}^2$ of $\mathbb{T}^2$ or restrict our attention to $[0,1]^2$ — thinking of the torus as the square $[0,1]^2$ with opposite edges identified. The attention function we define is a two times differentiable function $s : [0,1]^2 \to [0,1]$, as required by PID control, that remains C^2 under the identification of opposite edges.

The construction of s takes several steps. First, we consider just the attracting limit cycle of the reference field $\mathcal{R}$ (3.4) as encoding the essential behavior of the system. Collisions occur where this cycle crosses either of the two sections $\phi_1 = 0$ or $\phi_2 = 0$ of $\mathbb{T}^2$, thereby defining a characteristic sequence of collisions in the limiting behavior. We define a function s_1 from the limit cycle to the interval $[0,1]$ where $s_1 = 1$ near $\phi_1 = 0$ and $s_1 = 0$ near $\phi_2 = 0$ and varies smoothly between these extremes at all other points. Next, we extend s_1 to a switch function on the rectangle $[0,1] \times [-1,1]$ via an operation that is like a reverse deformation retraction. The result is a switch function on a two dimensional strip built around a copy of the limit cycle. The rectangle is then stretched, rotated and offset to lie between the lines $Ay = Bx$ and $Ay = Bx + 1$. Finally, this strip is wrapped around $\mathbb{T}^2$ to complete the construction. These steps are illustrated in Figure A.1. We describe the details next.

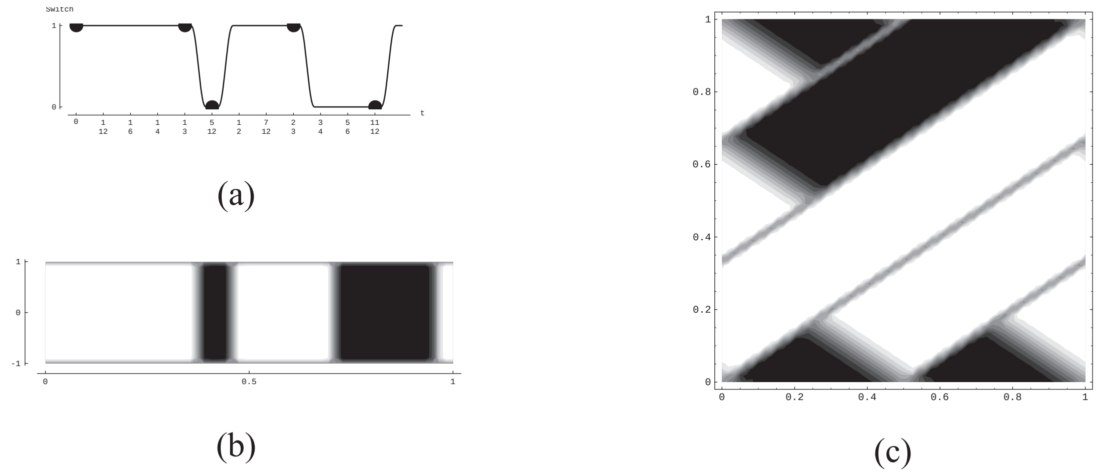


Figure A.1: The construction of an attention function for the case $A\!:\!B = 3\!:\!2$. (a) The attention function on an ideal version of the limit cycle of (3.4) with dots added to indicate when hits occur. (b) The limit cycle extended to $[0,1] \times [-1,1]$. (c) The strip in (b) wrapped around $\mathbb{T}^2$.

A.1 The Attention Function on the Limit Cycle

To construct s_1, we first examine some basic properties of the limit cycle and in particular the sequence of collisions that a given $A : B$ generate. To this end, parameterize the limit cycle in (3.4) by $t \mapsto (At, Bt + \frac{1}{2A})$ (taken modulo 1), $t \in \mathbb{R}$.

Definition A.1.1 *A* **hit point** *is a pair* (ϕ_1, ϕ_2) *where either* $\phi_1 \equiv 0 \pmod 1$ *or* $\phi_2 \equiv 0 \pmod 1$.

Hit points occur along the limit cycle when $At \equiv 0 \pmod 1$ or $Bt + \frac{1}{2A} \equiv 0 \pmod 1$ or equivalently when there is some $j \in \mathbb{Z}$ such that

$$t = \frac{j}{A} \;\text{ or }\; t = \frac{1}{B}(j - \frac{1}{2A}). \tag{A.1}$$

Property A.1.1 *On the limit cycle, points of the form* $(k,k) \in \mathbb{Z}^2$ *are not hit points. That is, hits do not occur simultaneously.*

Proof: Suppose $(k,k) \in \mathbb{Z}^2$ is a hit point on the limit cycle. Then for some t, $At = k$ and $Bt + \frac{1}{2A} = k$. Thus, substituting k/A for t in the second equation gives

$$B\frac{t}{k} + \frac{1}{2A} = k$$

or equivalently

$$Bk + \frac{1}{2} = k,$$

which is a contradiction. The term on the left hand side is not an integer, while the term on the right hand side is. □

It is convenient to define $\sigma = 2AB$, for it allows us an easy characterization of hit points, as the following property demonstrates.

Property A.1.2 *All hit points are of the form $(At, Bt + \frac{1}{2A})$ where $t = \frac{n}{\sigma}$ for some $n \in \mathbb{Z}$.*

Proof: Suppose $(At, Bt + \frac{1}{2A})$ is a hit point. Then either $At = k$ or $Bt + \frac{1}{2A} = k$ for some integer k. In the first case, $t = 2Bk/\sigma$ so that $n = 2Bk$. In the second case, $t = (2Ak - 1)/\sigma$ so that $n = 2Ak - 1$. □

We define the partial function $h : \mathbb{Z} \to \{0, 1, \perp\}$ to determine the kind of hit point $\frac{n}{\sigma}$ is, if it is a hit point at all. We let 1 correspond to a ball one hit, and 0 to a ball two hit. We correspond the special object $\perp$ (to mean *undefined*) to the case that neither ball is hit. Then we have

Proposition A.1.1 *Let $t = \frac{n}{\sigma}$ and $P = (At, Bt + \frac{1}{2A})$. Define the function $h : \mathbb{Z} \to \{0, 1, \perp\}$ by $h(n) = 1$ when P is a ball 1 hit point (i.e., $At \equiv 0$), $h(n) = 0$ when P is a ball 2 hit point (i.e., $Bt + \frac{1}{2A} \equiv 0$) and $h(n) = \perp$ when P is not a hit point. Then*

$$h(n) = \begin{cases} 1 \text{ if } \frac{nA}{\sigma} \in \mathbb{Z} \\ 0 \text{ if } \frac{(n+1)B}{\sigma} \in \mathbb{Z} \\ \perp \text{ otherwise } . \end{cases} \tag{A.2}$$

Note by Property A.1.1 that h is well defined. The proof is a simple rewriting of the conditions that $(At, Bt + \frac{1}{2A})$ be a hit point.

For those n such that $h(n) = \perp$, we have to decide which ball to attend to. An obvious choice is to have the paddle attend to the ball that will be hit next. So we define a new function $\tilde{h}$ recursively:

$$\tilde{h}(n) = \begin{cases} h(n) \text{ if } h(n) \neq \perp \\ \tilde{h}(n+1) \text{ otherwise.} \end{cases} \tag{A.3}$$

The function $\tilde{h}$ is the basis for the switch function on $\mathbb{T}^2$, which we are now prepared to build. Notice that $\tilde{h}$ is defined on the integers, but is described completely by its values on the set $\{0, 1, 2, ..., \sigma - 1\}$.

Property A.1.3 *$\tilde{h}(n + k\sigma) = \tilde{h}(n)$ for all integers k.*

Proof: First we show that $\frac{nA}{\sigma} \in \mathbb{Z}$ is equivalent to $\frac{(n+k\sigma)A}{\sigma} \in \mathbb{Z}$. Accordingly, if $\frac{nA}{\sigma} = j$ for some $j \in \mathbb{Z}$ then

$$\frac{(n + k\sigma)A}{\sigma} = j + kA \in \mathbb{Z}$$

since kA is an integer. Conversely, if $\frac{(n+k\sigma)A}{\sigma} = j$ for some $j \in \mathbb{Z}$ then

$$\frac{nA}{\sigma} = j - kA \in \mathbb{Z}.$$

By a similar argument, $\frac{(n+1)B}{\sigma} \in \mathbb{Z}$ is equivalent to $\frac{(n+k\sigma+1)B}{\sigma} \in \mathbb{Z}$. The desired result follows. □

Next, we define a function s_1 on the interval $[0, 1]$ by dividing $[0, 1]$ into σ subintervals $[\frac{n}{\sigma}, \frac{n+1}{\sigma}]$, $0 \leq n < \sigma$. The function $\tilde{h}$ tells us what to do at the endpoints of these intervals. Thus, $s_1(t)$ agrees with $\tilde{h}(\sigma t)$ when σt is an integer, or equivalently, when $\lfloor \sigma t \rfloor = \lceil \sigma t \rceil$. We just need to fill in the rest of the intervals. We will need C^2 step functions $up : \mathbb{R}^3 \to \mathbb{R}$ and $down : \mathbb{R}^3 \to \mathbb{R}$. The function $up(t, a, b)$ is used to fill in an interval $[a, b]$ wherein $\tilde{h}(a) = 0$ and $\tilde{h}(b) = 1$. It is thus 0 when $t \leq a$, between 0 and 1 when $a < t < b$ and 1 otherwise. Such a function can be constructed with polynomial splines of degree 6. *down* is defined similarly. The function *down* is used, for example, between the second and third hit points in Figure A.1(a). The smoothed function s_1 is then

$$s_1(t) = \begin{cases} 0 \text{ if } \tilde{h}(\lfloor \sigma t \rfloor) = \tilde{h}(\lceil \sigma t \rceil) = 0 \\ \\ 1 \text{ if } \tilde{h}(\lfloor \sigma t \rfloor) = \tilde{h}(\lceil \sigma t \rceil) = 1 \\ \\ up(t, \frac{\lfloor \sigma t \rfloor}{\sigma}, \frac{\lceil \sigma t \rceil}{\sigma}) \text{ if } \tilde{h}(\lfloor \sigma t \rfloor) = 0 \wedge \tilde{h}(\lceil \sigma t \rceil) = 1 \\ \\ down(t, \frac{\lfloor \sigma t \rfloor}{\sigma}, \frac{\lceil \sigma t \rceil}{\sigma}) \text{ if } \tilde{h}(\lfloor \sigma t \rfloor) = 1 \wedge \tilde{h}(\lceil \sigma t \rceil) = 0. \end{cases} \tag{A.4}$$

Notice that $s_1(0) = s_1(1)$ by Property A.1.3. This function is shown for the case $A\!:\!B = 3\!:\!2$ in Figure A.1(a).

A.2 Extending the Attention Function to $\mathbb{T}^2$

We further extend s_1 to the function $s_2 : [0,1] \times [-1,1] \to [0,1]$ defined by

$$s_2(x,y) = b(y)s_1(x) + \frac{1}{2}(1 - b(y))$$

where

$$b(y) = up(y, 0, \epsilon) \cdot down(y, 1-\epsilon, 1).$$

Here, ϵ is some small number that defines the width of the "borders" along $y = 1$ and $y = -1$. This function is shown is Figure A.1(b) as a contour plot. Note that if we form a cylinder or Möbius strip from the domain of s_2 by identifying the segment $x = 0$ with the segment $x = 1$, then s_2 is C^2 along the identification line. This follows from the cyclic nature of $\tilde{h}$ noted above.

We next take the domain of s_2, distort it and wrap it around $\mathbb{T}^2$ to complete the attention function. We wish for the line $y = 0$ in the domain of s_2 to be mapped to the limit cycle on T^2. Thus we define a map $f : [0,1] \times [-1,1] \to R^2$ by

$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} A & -\frac{1}{4B} \\ B & \frac{1}{4A} \end{bmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{2A} \end{pmatrix}. \tag{A.5}$$

As desired, $f(t,0)^T = (At, Bt + \frac{1}{2A})^T$.

The final step is to collapse the image of f down to $[0,1]^2$. For this we simply set $w(x,y) = (x \pmod 1, y \pmod 1)$ and obtain the desired switch function

$$s(\phi_1, \phi_2) = s_2 \circ f^{-1} \circ w^{-1}(\phi_1, \phi_2). \tag{A.6}$$

Since we designed s_2 so that for all x, $s_2(x, \pm 1) = \frac{1}{2}$, we are assured that along the "seams" of the wrapped s_2, s is still C^2. Notice that the paddle pays attention to neither ball along the seam. This is a that choice which we could have made differently. A contour plot of the case $A : B = 3 : 2$ is shown in Figure A.1(c).

Presently, although we know w^{-1} exists, we have only been able to find it for specific cases of A and B. A general formula has yet to be found. For practical purposes, however, it is not likely that arbitrary attention functions are useful. Rather, an attention function will be worked out for each of the basic cases $1:1$, $1:2$, $2:3$ and $2:5$ for example.

APPENDIX B

Single Hopper Return Map and Period of a Hop

To integrate the system (4.10), we change coordinates in the compression and decompression phase [14] so that the system

$$A = \begin{bmatrix} 0 & 1 \\ -\omega^2(1+\beta^2) & -2\omega\beta \end{bmatrix}$$

is in real canonical form [33]. The change of basis is given by

$$W = \begin{bmatrix} \omega\sqrt{1+\beta^2} & \frac{\beta}{\sqrt{1+\beta^2}} \\ 0 & \frac{1}{\sqrt{1+\beta^2}} \end{bmatrix}.$$

In the new coordinates we have the system

$$\begin{bmatrix} \dot{y} \\ \ddot{y} \end{bmatrix} = B \begin{bmatrix} y \\ \dot{y} \end{bmatrix}, \qquad B = WAW^{-1} = \omega \begin{bmatrix} -\beta & 1 \\ -1 & -\beta \end{bmatrix}.$$

We define energy and angle to be

$$E_c \doteq [x, \dot{x}]W^T W \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

$$\theta_c \doteq tan^{-1}\left(\frac{\dot{y}}{y}\right) = tan^{-1}\left(\frac{(1+\beta^2)^{-1/2}\dot{x}}{\omega(1+\beta^2)^{1/2}x + \beta(1+\beta^2)^{-1/2}\dot{x}}\right).$$

The subscript c denotes "compression." In these coordinates, the compression phase becomes

$$\begin{aligned} \dot{E}_c &= -2\omega\beta E_c \\ \dot{\theta}_c &= -\omega. \end{aligned}$$

A similar expression is obtained for θ_d and E_d where the d stands for "decompression":

$$\begin{aligned}\dot{E}_d &= -2\omega\beta E_d \\ \dot{\theta}_d &= -\omega\tau.\end{aligned}$$

Thus, both θ_c and θ_d have a constant rate of change (and decrease in time).

Now, starting at a point $(x_b, 0)$ corresponds to starting at a point (E_0, π). The lift-off point is (E_l, θ_l) and the touchdown point is $(E_{td}, \theta_t d)$. Since energy is conserved in flight, we have that $(E_{td}, \theta_{td}) = (E_l, \theta_l - \pi)$. Now, integrating (B.1) gives

$$E_l = E_0 e^{2\beta_2(\theta_l - \pi)}$$

and we know $E_{td} = E_l$. Finally, let E_b be the energy at the next bottom point. Then

$$\begin{aligned}E_b &= E_{td}e^{2\beta(\theta_b - \theta_{td})} = E_{td}e^{2\beta(-\pi-(\theta_l-\pi))} = E_l e^{-2\beta\theta_l} \\ &= E_0 e^{2\beta_2(\theta_l-\pi)}e^{-2\beta\theta_l} = E_0 e^{2[\theta_l(\beta_2-\beta)-\beta_2\pi]}.\end{aligned}$$

Now, $E_0 = \omega_2^2(1+\beta_2^2)x_b^2$ and $E_b = \omega^2(1+\beta^2)x_{b,next}^2$ and $\theta_l = \tan^{-1}(1/\beta_2)$. Therefore,

$$x_{b,next} = \frac{\omega_2\sqrt{1+\beta_2^2}}{\omega\sqrt{1+\beta^2}} x_b e^{\tan^{-1}(1/\beta_2)(\beta_2-\beta)-\beta\pi}. \tag{B.1}$$

Substituting $\beta_2 = \beta$ and $\omega_2 = \omega\tau$ with $\tau = (1-k_b)e^{\beta\pi}/(1-x_b)$ results in (4.10), denoted by f.

The return map f has the two fixed points 0 and k_b. Assuming $k_b < 0$, the derivative of f at 0 is $1 - k_b > 1$ and, thus, 0 is an unstable fixed point of f. At k_b, the derivative of f is $1/(1-k_b) < 1$ and, thus, k_b is a stable fixed point of f. Since there is a unique stable fixed point of f, there is a unique, closed stable orbit of the system given by (4.10) which passes through the point $(k_b, 0)$.

Derivation of the Period of a Hop

Let t_0, t_l, t_{td} and t_b be the initial time at bottom, the lift-off time, the touchdown time and the next bottom time. Then $t_l = t_d$ is the time of decompression,$t_{td} = t_d + t_f$ is the sum of the decompression time and the flight time, and $t_b = t_d + t_f + t_c$.

Now, integrate $\dot{\theta}_d = -\omega_2$ to get $\theta_l = \theta_0 - \omega_2 t_d$ so that

$$t_d = \frac{1}{\omega_2}(\theta_0 - \theta_l) = \frac{1}{\omega\tau}\left[\pi - \tan^{-1}(1/\beta)\right].$$

We next need the velocity $\dot{x}_l$ at lift-off. This can be found using the equation

$$E_l = (0, \dot{x}_l) W^T W (0, \dot{x}_l)^T = \dot{x}_l^2.$$

Thus,

$$\dot{x}_l = \sqrt{E_l} = -\omega\tau\sqrt{1+\beta^2} x_b e^{\beta(\pi-\theta_l)}.$$

Now integrating the flight phase gives that

$$t_f = \frac{2\dot{x}_l}{\gamma} = -\frac{2}{\gamma}\omega\tau\sqrt{1+\beta^2} x_b e^{\beta(\pi-\theta_l)}.$$

Lastly, integrating $\dot{\theta}_c = -\omega$ from θ_{td} to $-\pi$ and solving for t_c gives

$$t_c = \frac{1}{\omega}\tan^{-1}(1/\beta).$$

Summing the three times and substituting the value for τ given in the main text yields the desired result, (4.11).

APPENDIX C

Proof of Proposition 4.5.4

We describe the salient points of the proof of this theorem. Essentially, we linearize F and show that the linearized system is stable at (T^*, π, T^*). To compute the Jacobian of the map F, first define

$$\begin{aligned} a &\triangleq (\pi - \theta_l)e^{\beta\pi} \\ b &\triangleq \frac{1}{\gamma}2\omega e^{\beta\theta_l}\sqrt{1+\beta^2} \\ c &\triangleq \theta/\omega \\ \delta &\doteq \frac{k_b k_s \pi b}{(1-k_b)(a - bk_b + c)} \end{aligned}$$

Straightforward computation of partial derivatives yields that the Jacobian evaluated at (T, π, T) is equal to

$$\begin{pmatrix} \frac{1}{1-k_b} & \frac{T\delta}{\pi} & 0 \\ \frac{\pi(2+\delta)}{T(1-k_b)} & 1 + 3\delta + \delta^2 & \frac{\pi(2-k_b+\delta(1-k_b))}{T(k_b-1)} \\ \frac{\delta}{k_b-1} & -\frac{T\delta(1+\delta)}{\pi} & \frac{1}{1-k_b} + \delta \end{pmatrix}. \tag{C.1}$$

Finding the characteristic polynomial of (C.1) and substituting (4.22) for k_s gives

$$-\lambda^3 + \xi_1\lambda + \xi_0. \tag{C.2}$$

Where

$$\xi_0 = \frac{1}{(k_b-1)^2} \text{ and}$$

$$\xi_1 = \frac{k_b}{(k_b-1)^2}(6 - 7k - 4\sqrt{1 - 4k_b + 3k_b^2}).$$

We may now show the following:

Lemma C.0.1 *The roots of (C.2) all have magnitude less than one whenever k_b is negative.*

Proof: Suppose ρ_1, ρ_2 and ρ_3 are the roots of (C.2). Then

$$(\lambda - \rho_1)(\lambda - \rho_2)(\lambda - \rho_3) = \lambda^3 - \xi_1 \lambda - \xi_0.$$

Thus,

$$\begin{aligned} \rho_1 + \rho_2 + \rho_3 &= 0 \\ \rho_1\rho_2 + \rho_1\rho_3 + \rho_2\rho_3 &= -\xi_1 \\ \rho_1\rho_2\rho_3 &= \xi_0. \end{aligned} \tag{C.3}$$

Now, it can be shown that when $k_b < 0$ the coefficients of (C.2) satisfy the conditions $0 < \xi_0 < 1$ and $-1 < \xi_1 < 0$ giving the following conditions on the roots

$$\begin{gathered} \rho_1 + \rho_2 + \rho_3 = 0 \\ 0 < \rho_1\rho_2 + \rho_1\rho_3 + \rho_2\rho_3 < 1 \\ 0 < \rho_1\rho_2\rho_3 < 1 \end{gathered} .$$

Using these conditions it is straightforward to show that two of the roots are complex conjugates, the other is real and negative and all have magnitude less than one.

Now, since the eigenvalues of (C.1) all have magnitudes less than one, we can conclude that (T^*, π, T^*) is a stable fixed point of the system (4.21). $\square$

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] M. Abadi and L. Lamport. Composing specifications. *ACM Transactions on Programming Languages and Systems*, 15(1):75–132, January 1993.

[2] R. Altendorfer, U. Saranli, H. Komsuoglu, D. E. Koditschek, B. Brown, D. McMordie, N. Moore, M. Buehler, and R. Full. Evidence for spring loaded inverted pendulum running in a hexapod robot. In *International Symposium for Experimental Robotics*, Honolulu, HI, December 2000.

[3] R. C. Arkin. Motor schema based navigation for a mobile robot. *Proceedings of the International Conference on Robotics and Automation*, pages 264–271, 1987.

[4] V. I. Arnold. *Mathematical Methods of Classical Mechanics*. Springer, 2nd edition, 1991. Translated by K. Vogtmann and A. Weinstein.

[5] J. C. M. Baeten and W. P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge Univ. Press, 1990.

[6] R. Beer, R. Quinn, H. Chiel, and R. Ritzmann. Biologically inspired approaches to robotics. *Communications of the ACM*, 40(3):31–38, March 1997.

[7] E. S. Bizzi, S. F. Giszter, E. Loeb, F. A. Mussa-Ivaldi, and P. Saltiel. Modular organization of motor behaviors in the frog's spinal chord. *Trends in Neuroscience*, 18:442–446, 1995. Review.

[8] A. M. Bloch, P. S. Krishnaprasad, J. E. Marsden, and R. Murray. Nonholonomic mechanical systems with symmetry. *Archive for Rational Mechanics and Analysis*, 136:21–99, 1996.

[9] R. W. Brockett. Asymptotic stability and feedback stabilization. In R. W. Brockett, R. S. Millman, and H. J. Sussman, editors, *Differential Geometric Control*, chapter 3, pages 181–191. Birkhäuser, 1983.

[10] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2:14–23, 1986.

[11] R.A. Brooks. Intelligence without representation. In *Proceedings of the Workshop on the Foundations of AI*, Endicott House, 1987.

[12] M. Bühler, D. E. Koditschek, and P. J. Kindlmann. Robotics in intermittent dynamical environments. Technical Report 8812, Yale University, 1988. Revised 1989.

[13] M. Bühler, D. E. Koditschek, and P. J. Kindlmann. A family of control strategies for intermittent dynamical environments. *IEEE Control Systems Magazine*, 10(2):16–22, 1990.

[14] M. Bühler, D. E. Koditschek, and P.J. Kindlmann. Planning and control of robotic juggling and catching tasks. *International Journal of Robotics Research*, 13(2):101–118, April 1994.

[15] R. Burridge, A. Rizzi, and D. E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *International Journal of Robotics Research*, 18(6):534–d55, 1999.

[16] A. Calvitti and R. D. Beer. Analysis of a distributed model of leg coordination. I. Individual Coordination Mechanisms. *Biological Cybernetics*, 82(3):197–206, 2000.

[17] A. G. Cass and L. J. Osterweil. Design guidance through the controlled application of constraints. In *Proceedings of the 10th International Workshop in Software Specification and Design (IWSSD 10)*, pages 195–199, San Diego, CA, November 2000.

[18] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[19] K. M. Chandy and B. A. Sanders. Reasoning about program composition. http://www.cise.ufl.edu/∼sanders/pubs/index.html.

[20] M. Charpentier and K. M. Chandy. Towards a compositional approach to the design and verification of distributed systems. In J. Wing, J. Woodcock, and J. Davies, editors, *World Congress on Formal Methods in the Development of Computing Systems (FM'99)*, volume 1708 of *Lecture Notes in Computer Science*, pages 570–589. Springer-Verlag, September 1999.

[21] A. H. Cohen, P. J. Holmes, and R. H. Rand. The nature of the coupling between segmental oscillators of the lamprey spinal generator for locomotion: A mathematical model. *J. Math. Biology*, 13:345–369, 1982.

[22] A. H. Cohen, S. Rossignol, and S. Grillner (eds.). *Neural Control of Rhythmic Movements in Vertebrates*. Wiley Inter-Science, NY, 1988.

[23] F. Commoner, A.W. Holt, S. Even, and A. Puneli. Marked directed graphs. *Journal of Computer and System Sciences*, 5:511–523, 1971.

[24] J. C. Corbett and G. S. Avrunin. Towards scalable compositional analysis. In *Proc. Second ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 53–61, December 1994.

[25] H. Cruse. What mechanisms coordinate leg movement in walking arthropods? *Trends in Neurosciences*, 13:15–21, 1990.

[26] F. Delcomyn. Neural basis of rhythmic behavior in animals. *Science*, 1980.

[27] B. R. Donald, J. Suh, R. B. Darling, K.-F. Bhringer, H. Baltes, and G. Kovacs. Cmos integrated organic ciliary actuator arrays for general-purpose micromanipulation tasks. *Journal of Microelectromechanical Systems*, 8(4):483–496, December 1999.

[28] G.B. Ermentrout and N. Kopell. Inhibition-produced patterning in chains of coupled nonlinear oscillators. *SIAM Journal of Applied Mathematics*, 54:478–507, 1994.

[29] R. J. Full and D. E. Koditschek. Templates and anchors: Neuromechanical hypotheses of legged locomotion. *Journal of Experimental Biology*, 202(23):3325–3332, 1999.

[30] G. Gabrielli and T. H. von Karman. What price speed? *Mechanical Engineering*, 72(10), 1950.

[31] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer-Verlag, New York, 1983.

[32] G. Hawker and M. Buehler. Quadruped trotting with passive knees — design, control, and experiments. In *Proc. of the IEEE Int. Conf. Robotics and Automation*, pages 3046–3051, San Francisco, CA, April 2000.

[33] M.W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems, and Linear Algebra*. Academic Press, 1974.

[34] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–583, October 1969.

[35] C. A. R. Hoare. *Communicated Sequential Processes*. Series in Computer Science. Prentice Hall, 1985.

[36] J. J. Hu. *Stable Locomotion Control of Bipedal Walking Robots: Synchronization with Neural Oscillators and Switching Control*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2000.

[37] R. Janicki. Nets, sequential compositions and concurrency relations. *Theoretical Computer Science*, 29:87–121, 1984.

[38] O. Khatib. Real time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–99, 1986.

[39] E. Klavins. Automatic compilation of concurrent hybrid factories from product assembly specifications. In *Hybrid Systems: Computation and Control Workshop, Third International Workshop*, Pittsburgh, PA, 2000.

[40] E. Klavins, D. E. Koditschek, and R. Ghrist. Toward the regulation and composition of cyclic behaviors. In *Proceedings of the Fourth International Workshop on the Algorithmic Foundations of Robotics 2000*, Dartmouth, NH, 2000.

[41] E. Klavins and D.E. Koditschek. A formalism for the composition of loosely coupled robot behaviors. Technical Report CSE-TR-412-99, University of Michigan, 1999.

[42] E. Klavins and D.E. Koditschek. A formalism for the composition of concurrent robot behaviors. In *Proceedings of the IEEE Conference on Robotics and Automation*, 2000.

[43] E. Klavins, H. Komsuoglu, D. E. Koditschek, and R. J. Full. The role of reflexes versus central pattern generators in dynamical legged locomotion. In *Neurotechnology for Biomimetic Robots*. MIT Press, 2001. In Press.

[44] D. E. Koditschek. The application of total energy as a lyapunov function for mechanical control systems. In J. Marsden, P. S. Krishnaprasad, and J. Simo, editors, *Control Theory and Multibody Systems*, volume 97 of *AMS Series in Contemporary Mathematics*, pages 131–158. 1989.

[45] D. E. Koditschek. Computational neuromechanics: Programming work in biological and machine systems. Available at http://ai.eecs.umich.edu/CNM/, 1999.

[46] D. E. Koditschek and M. Bühler. Analysis of a simplified hopping robot. *International Journal of Robotics Research*, 10(6):587–605, December 1991.

[47] N. Kopell. Chains of coupled oscillators. In M. Arbib, editor, *Handbook of Brain Theory and Neural Networks*, pages 178–183. MIT Press, Cambridge, MA, 1995.

[48] J. Košecká. *A Framework for Modeling and Verifying Visually Guided Agents: Design, Analysis and Experiments.* PhD thesis, University of Pennsylvania, March 1996.

[49] L. Lamport. Composition: A way to make proofs harder. In W. de Roever, H. Langmaack, and A. Puneli, editors, *Compositionality: The significant difference (Proceedings of the COMPOS'97 Symposium*, volume 1563, pages 402–423. 1998.

[50] E. W. Large, H. I. Christensen, and R. Bajcsy. Scaling the dynamic approach to planning and control: Competition among behavioral contraints. *International Journal of Robotics and Automation*, 18(1):37–58, January 1999.

[51] J. P. LaSalle. *Stability by Liapunov's direct method.* Mathematics in science and engineering. v. 4. New York, Academic Press, 1961.

[52] M. A. Lewis, R. Etienne-Cummings, A.H. Cohen, and M. Hartmann. Toward biomorphic control using custom avlsi chips. In *Proceedings of the 2000 International Conference on Robotics and Automation*, San Francisco, April 2000.

[53] W. Lohmiller and J. E. Slotine. On contraction analysis for non-linear systems. *Automatica*, 34(6):683–696, 1998.

[54] T. Lozano-Perez and M. T. Mason. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1):3–24, Spring 1984.

[55] T. Lozano-Perez, M.T. Mason, and R.H. Taylor. Automatic synthesis of fine-motion strategies for robots. *The International Journal for Robotics Research*, 3(1):3–23, 1984.

[56] N. Lynch. *Distrbuted Algorithms.* Morgan Kaufman, 1996.

[57] R. Milner. *A Calculus of Communcating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer Verlag, New York, 1980.

[58] F. A. Mussa-Ivaldi. Nonlinear force fields: A distributed system of control primitives for representing and learning movements. In *Proceedings of the IEEE Int. Symp. on Computational Intelligence in Robotics and Automation*, pages 84–90, July 1997.

[59] J. Nakanishi, T. Fukuda, and D. E. Koditschek. A brachiating robot controller. *IEEE Transactions on Robotics and Automation*, 2000. (to appear).

[60] G. N. Orlovsky, T. G. Deliagina, and S. Grillner. *Neural Control of Locomotion.* Oxford University Press, New York, 1999.

[61] J. Ostrowski, J. P. Desai, and V. Kumar. Optimal gait selection for nonholonomic locomotion systems. *International Journal of Robotics Research*, 19(3):225–237, 1998.

[62] J. P. Ostrowski and K. A. McIsaac. A framework for steering dynamic robotic locomotion systems. In B. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: 2000 WAFR*, pages 221–232. AK Peters, 2001.

[63] E. Park, D. M. Tilbury, and P.P. Khargonekar. Modular logic controller for maching systems: Formal representation and performance analysis using petri nets. *IEE Transactions on Robotics and Automation*, (to appear).

[64] K. Pearson. The control of walking. *Scientific American*, 235(6):72–86, December 1973.

[65] G. A. Pratt and J. Nguyen. Distributed synchronous clocking. *IEEE Transactions on Parallel and Distributed Systems*, 6(3):314–328, 1995.

[66] M.H. Raibert. *Legged Robots that Balance.* The MIT Press Series in Artificial Intelligence. MIT Press, 1986.

[67] M.H. Raibert, H.B. Brown, and M. Chepponis. Experiments in balance with a 3D one-legged hopping machine. *International Journal of Robotics Research*, 3:75–92, 1984.

[68] J.H. Reif and H. Wang. Social potential fields: A distributed behavioral control for autonomous robots. *Robotics and Autonomous Systems*, 27(3):171–194, May 1999.

[69] W. Reisig. *Petri Nets: An Introduction.* Springer Verlag, 1985.

[70] E. Rimon and D.E. Koditschek. Exact robot navigation using artificial potential fields. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, October 1992.

[71] A. A. Rizzi, J. Gowdy, and R. L. Hollis. Distributed programming and coordination for agent-based modular automation. In *The Ninth International Symposium of Robotics Research*, Snowbird, UT, October 1999.

[72] A.A. Rizzi and D. E. Koditschek. Further progress in robot juggling: Solvable mirror laws. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Atlanta, GA, May 1993.

[73] A.A. Rizzi, L.L. Whitcomb, and D.E. Koditschek. Distributed real-time control of a spatial robot juggler. *IEEE Computer*, 25(5):12–26, May 1992.

[74] U. Saranli, M. Buehler, and D.E. Koditschek. Design, modeling and preliminary control of a compliant hexapod robot. In *Proceedings of the IEEE Conference on Robotics and Automation*, 2000.

[75] U. Saranli, W.J. Schwind, and D.E. Koditschek. Toward the control of a multi-jointed monoped runner. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 2676–2682, 1998.

[76] G. Schöner, M. Dose, and C. Engels. Dynamics of behavior: Theory and applications for autonomous robot architectures. *Robotics and Autonomous Systems*, 16:213–245, 1995.

[77] Z. Shao, C. League, and S. Monnier. Implementing typed intermediate languages. In *Proc. 1998 ACM SIGPLAN International Conference on Functional Programming (ICFP'98)*, pages 313–323, Baltimore, MD, September 1998.

[78] D. Tennenhouse. Proactive computing. *Communications of the ACM*, 43(5):43–50, 2000.

ABSTRACT

Decentralized Phase Regulation
of Cyclic Robotic Systems

by

Eric Klavins

Co-Chairs: Daniel E. Koditschek and William C. Rounds

Control algorithms are difficult to scale up to large decentralized systems because of potentially complex couplings between components. As a result, many such systems function conservatively – damping out most of their energy and moving slowly or in a start-stop fashion. Synthesis tools for dynamical behaviors that admit a modular, bottom-up approach attempt to address this problem. Without such tools, the full potential of modern actuators, sensors and computing power likely cannot be realized, dooming robots (and more generally, physically situated computing systems) to a clumsy and inefficient future.

This dissertation describes efforts to provide a formal basis for designing and verifying decentralized control algorithms for robotic systems such as factories, dynamic manipulators, hoppers and walkers. The methods are based on *synthesis* and, particularly, composition. The idea behind the compositional approach is simple. A designer should not have to start from scratch when building complicated systems. Instead knowledge of how to make a machine do two tasks separately should be leveraged to make the machine do the two tasks in some combination.

To avoid the complexity of arbitrary couplings, we consider systems that may be decomposed in very regular ways. In particular, we focus on methods for coordinating multiple cyclic behaviors, based on defining ideal, model dynamical systems

called *reference fields*. In using reference fields to control coupled cyclic systems, it is assumed that each cyclic system can be continuously actuated. However, many tasks in robotics, such as "juggling" and hopping, allow only intermittent control. Thus, we also show that reference fields can be used as the basis for controlling these more complicated tasks. Finally, reference fields are applied to the control of a six-legged, scampering robot and the performance of the approach is examined with respect to the speed and power consumption of the robot.