

Exploiting Spatial and Temporal Flexibility for Plan Execution of Hybrid, Under-actuated Systems

Andreas G. Hofmann and Brian C. Williams

Computer Science and Artificial Intelligence Lab, MIT
32 Vassar St. rm. 32-275
Cambridge, MA 02139
hofma@csail.mit.edu, williams@mit.edu

Abstract

Robotic devices, such as rovers and autonomous spacecraft, have been successfully controlled by plan execution systems that use plans with temporal flexibility to dynamically adapt to temporal disturbances. To date these execution systems apply to discrete systems that abstract away the detailed dynamic constraints of the controlled device. To control dynamic, under-actuated devices, such as agile bipedal walking machines, we extend this execution paradigm to incorporate detailed dynamic constraints.

Building upon prior work on dispatchable plan execution, we introduce a novel approach to flexible plan execution of hybrid under-actuated systems that achieves robustness by exploiting spatial as well as temporal plan flexibility. To accomplish this, we first transform the high-dimensional system into a set of low dimensional, weakly coupled systems. Second, to coordinate these systems such that they achieve the plan in real-time, we compile a plan into a *concurrent timed flow tube description*. This description represents all feasible control trajectories and their temporal coordination constraints, such that each trajectory satisfies all plan and dynamic constraints. Finally, the problem of runtime plan dispatching is reduced to maintaining state trajectories in their associated flow tubes, while satisfying the coordination constraints. This is accomplished through an efficient local search algorithm that adjusts a small number of control parameters in real-time. The first step has been published previously; this paper focuses on the last two steps. The approach is validated on the execution of a set of bipedal walking plans, using a high fidelity simulation of a biped.

Introduction

Past work in qualitative reasoning has produced methods for controlling dynamic systems that are distinguished in their use of qualitative descriptions of dynamics to control robustly over large regions of state space [Kuipers and Ramamoorthy, 2001; Hofbaur, 1999]. In particular, the qualitative concept of flow tubes [Bradley and Zhao, 1993; Frazzoli, 2001] explicitly defines the control regime that is

handled robustly. With this approach, compile time methods identify bundles of trajectories, called *flow tubes*, that navigate the system to a desired equilibrium point.

One limitation of the approach is the exhaustive state space exploration performed by these methods. Consequently, they have only been applied to relatively low-dimensional systems, and have not scaled well to high-dimensional systems at the level of complexity, for example, of walking robots (Fig. 1).

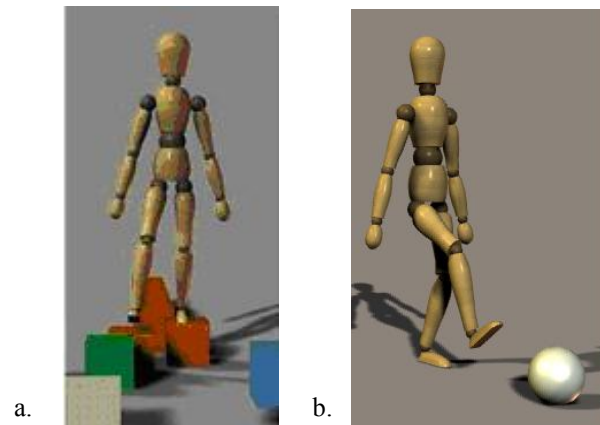


Fig. 1 – a. Walking on difficult terrain; b. kicking a ball

A second limitation is that the goals achieved by these qualitative controllers are restricted to simple set point regions; this type of description is not adequate for specifying complex tasks. Such complex tasks are distinguished by the execution of sequences of concurrent activities that are coordinated through timing constraints. Furthermore, while flow tubes provide state-space trajectory flexibility, they do not provide a representation of temporal flexibility.

Conversely, work on temporally flexible plan execution deals with the execution of complex tasks with many of these attributes [Tsamardinos et al., 1998; Muscettola et al., 1998]. This approach achieves robustness to disturbances through compile-time methods that make explicit the family of activity execution schedules that satisfy a plan's temporal constraints, and

then dynamically updating this family of schedules in response to disturbances.

However, this approach ignores the continuous dynamics of the underlying plant, and assumes that plan activities can be started and finished at arbitrary times, as long as these times are within the bounds specified by the temporal constraints of the plan. This is not valid for under-actuated dynamic systems, like walking bipeds, because the state of these systems is modified continuously through their second derivative (e.g., acceleration) resulting in inertia that can delay the achievement of a desired goal region.

Furthermore, both flow tube based control systems, and temporally flexible plan execution systems, as well as STRIPS planners in general, rely heavily on the notion of equilibrium points. Equilibrium points are points in state space where the system is at rest, and remains at rest if there is no disturbance or control action. This emphasis on equilibrium points makes these methods unsuitable for an important class of problems in which high performance is desired from an under-actuated plant.

Agile bipedal walking on difficult terrain, shown in Fig. 1a, is an important example of this class of problem. Such walking is achieved through appropriate velocity control, with the emphasis on limit cycle stability, rather than on achieving an equilibrium point. In fact, agile walking, and similar high performance tasks, is characterized by a lack of equilibrium points; successful execution is defined by achieving a sequence of goal regions that do not contain any equilibrium points. A related example is kicking a soccer ball, as shown in Fig. 1b. Stepping movement must be synchronized with ball movement so that the kick happens when the ball is close enough.

For such systems, traditional concepts of stability are meaningless. Therefore, flow tube systems that explore state space in search of equilibrium points are not useful for this type of application. Similarly, the ability of a temporally flexible plan execution system to set activity execution times relies on the ability of the system to idle in an equilibrium point for an arbitrary period of time. This is not guaranteed in an under-actuated system. Thus, neither the flow tube systems, nor the activity execution systems, deal adequately with applications where there are few or no equilibrium points.

Key Innovations of Approach

In this paper we present *Chekhov*, a system for robust, task-level control of high-performance, under-actuated systems with continuous dynamics. *Chekhov* achieves responsiveness and robustness through a compiled representation, called a *qualitative control plan* (QCP). This type of plan unifies the seemingly disparate representations used in flow tube control and activity execution systems, resulting in a system that combines the strengths of each.

Similar to flow tubes, a QCP maintains a representation of bundles of feasible trajectories. However, unlike flow tubes, a QCP addresses the problem of a high dimensional

state space by factoring the state space into a set of concurrently operating single-input, single-output 2nd-order systems, and by representing feasible trajectories with a set of concurrent flow tubes related through timing constraints. Similar to activity execution systems, the flow tubes in a QCP are associated with sequences of concurrent activities, rather than with a single setpoint; successful execution of the activities corresponds to successful execution of the plan. Also similar to activity execution systems, the temporal constraints of a QCP are compiled into sets of feasible schedules, and are updated dynamically using a simple dispatching algorithm. However, the approach is distinguished in that the QCP incorporates the temporal constraints imposed by the plant dynamics.

Similar to both flow tube control and activity execution systems, a QCP is executed robustly using a combination of off-line compilation methods, and efficient on-line dispatching that adapts dynamically to disturbances. As with flow tube systems, the dispatcher selects control values that maintain the system along a feasible trajectory within a flow tube. As with activity execution systems, the QCP dispatcher schedules the execution times of activities according to the compiled temporal constraints.

However, the approach is unique in that it also avoids the reliance on equilibrium points of the other two methods by supporting sequences of goal regions without such points. Because the temporal constraints imposed by the plant dynamics are included, the system knows how long it is able to reside in a goal region that does not contain an equilibrium point.

Compiled Model-based Executive for Under-actuated Systems

We use a *model-based executive* [Williams and Nayak, 1997; Leaute and Williams, 2005] to interpret plan goals, monitor plant state, and compute control actions, as shown in Fig. 2.

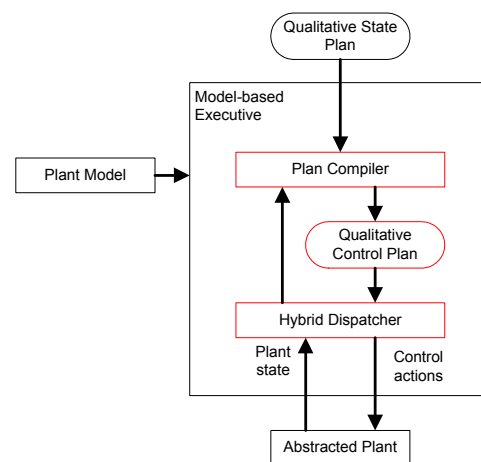


Fig. 2 – Model-based executive

The inputs to the model-based executive include a *Qualitative State Plan (QSP)*, which specifies state-space and temporal requirements, the current plant state, and a plant model. The outputs are control actions for the *abstracted plant*. As shown in Fig. 2, the executive consists of a plan compiler, which is the off-line component, and a dispatcher, which is the on-line component. The plan compiler generates a QCP corresponding to the QSP. The QCP contains the flow tubes and the dispatchable graph. It is executed by the dispatcher, which schedules activity start times using a method similar to that used in the plan execution systems, and executes activities by keeping trajectories in their flow tubes.

Abstracted Plant

The abstracted plant is a set of linear, 2nd-order single-input single-output (SISO) systems, as shown in Fig. 3. Each SISO system is controlled by a *proportional-differential (PD)* control law [Ogata, 1982], which is of the form

$$\ddot{y} = k_p(y_{set} - y) + k_d(\dot{y}_{set} - \dot{y}) \quad (1)$$

where y_{set} and \dot{y}_{set} are position and velocity setpoints, and k_p and k_d are proportional and differential gains.

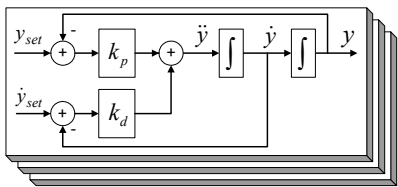


Fig. 3 – Abstracted plant is a set of SISO systems

Definition 1 (SISO System): A single-input single-output (SISO) system is a tuple, $\langle y_{set}, \dot{y}_{set}, k_p, k_d \rangle$. Given an SISO system, S , a plant trajectory of S , $traj(S)$, is a function of time, $y(t)$, that satisfies Eq. (1), where the control parameters in Eq. (1) are given by S . The SISO system may, optionally, have a set of constant constraints on the control parameters in S . These constraints are of the form

$$y_{set_min} \leq y_{set} \leq y_{set_max}$$

$$\dot{y}_{set_min} \leq \dot{y}_{set} \leq \dot{y}_{set_max}$$

$$k_{p_min} \leq k_p \leq k_{p_max}$$

$$k_{d_min} \leq k_d \leq k_{d_max}$$

These constraints may vary in time, according to discrete mode of the plant.

The modes of the plant, and associated operation constraints, are specified by activities in the QSP, as discussed in the next section. For a walking biped, discrete modes are defined by the *base of support*, the convex hull of points in contact with the ground. The base of support changes discontinuously with each step. The horizontal

balance force that can be applied to the biped's *center of mass (CM)* is limited by the base of support, as shown in Fig. 4. The *ground reaction force vector*, as shown in Fig. 4. The *ground reaction force vector* represents the overall force exerted on the CM. This vector emanates from a point on the ground called the *zero moment point (ZMP)* [Vukobratovic and Juricic, 1969], and points directly at the CM if the moment about the CM is zero [Popovic, et al., 2004]. Thus, the horizontal force on the CM can be adjusted by moving the ZMP. However, this horizontal force is limited because the ZMP is required to be within the base of support.

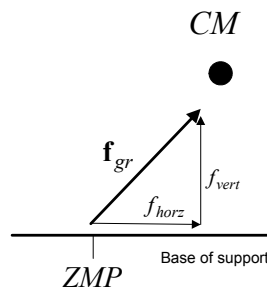


Fig. 4 – The ground reaction force vector, f_{gr} , emanates from the ZMP and exerts a horizontal force on the CM

For normal walking, the horizontal force on the CM is well approximated by

$$f_{horz} = k_s ZMP$$

, where k_s is an empirically determined spring constant [Popovic, et al., 2004]. In terms of Def. 1, y_{set} is the ZMP position, and k_p is k_s , scaled by the total mass of the biped. Thus, a limit on ZMP position represents a limit on y_{set} of the form given in Def. 1.

In some applications, the actual plant is of the form shown in Fig. 3 and Def. 1. More typically, as is the case with a walking robot, the actual plant is nonlinear and tightly coupled. In this case, we linearize and decouple the actual plant using an enhanced feedback linearizing controller [Hofmann, et. al., 2004; Slotine and Li, 1991], which transforms the actual plant into the required abstracted form.

Qualitative State Plan

A QSP specifies the desired behavior of the plant in terms of allowed state trajectories. We use a *qualitative state* to specify desired state-space goal regions, and temporal constraints to specify time ranges by which the state space goals must be achieved. For a walking biped, a qualitative state indicates which feet are on the ground, and includes constraints on foot position. It may also include constraints on the biped's center of mass position and velocity. A sequence of qualitative states represents intermediate goals that lead to the final overall plan goal, as shown in Fig. 5.

A qualitative state plan has a set of *activities* representing constraints on desired state evolution. In Fig. 5, the activity *left foot ground 1* is for the left foot, *right foot ground 1*, is for the right foot, and CM1 – 4 are for the center of mass. Every activity starts and ends with an *event*, represented by a circle in Fig. 4. Events in this plan relate to behavior of the stepping foot. Thus, a *toe-off* event represents the stepping foot lifting off the ground, and a *heel-strike* event represents the stepping foot landing on the ground. Events define the boundaries of qualitative states. For example, the right toe-off event defines the end of the first qualitative state (double support), and the beginning of the second qualitative state (left single support).

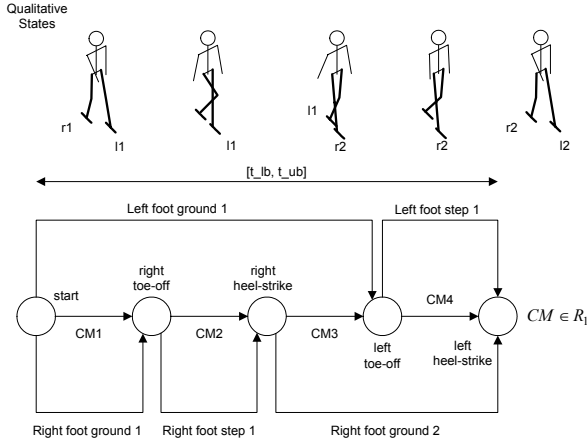


Fig. 5 - Example qualitative state plan for walking gait cycle. Circles represent events, and horizontal arrows represent activities.

The qualitative state plan in Fig. 5 has a temporal constraint between the start and finish events. In addition to temporal constraints, qualitative state plans can include required initial and goal regions for activities. In Fig. 5, the goal region constraint $CM \in R_1$ represents the requirement that the CM trajectory must be in region R_1 in order for the activity to finish successfully.

Each activity has an associated SISO system in the abstracted plant. An activity may specify constraints on the control parameters in the SISO system, corresponding to actuation limits. We assume that these constraints are constant limits; more general types of constraints are possible, but are beyond the scope of this discussion. In Fig. 5, the activities CM1 – CM4, representing CM movement, have different actuation limits. This is due to the discontinuous changes in the base of support resulting from the foot contact events.

Definition 2 (QSP): A qualitative state plan (QSP) is a tuple $\langle E, A, TC \rangle$, where E is a set of events, A is a set of activities, (Def. 3), and TC is a set of externally imposed temporal constraints on the events (Def. 4). An event, ev , represents a point in time.

Definition 3 (Activity): An activity is a tuple $\langle ev_s, ev_f, R_{op}, R_{init}, R_{goal}, S, A_{next} \rangle$, where ev_s and ev_f are activity start and finish events, R_{op} is a set of actuation constraints, R_{init} and R_{goal} are required state-space regions for start and finish of the activity, S is the SISO system (Def. 1), associated with the activity, and A_{next} is an optional successor activity. R_{op} specifies constant limits on the SISO control parameters.

Definition 4 (Temporal Constraint): A temporal constraint is a tuple $\langle ev_1, ev_2, l, u \rangle$, where ev_1 and ev_2 are events (Def. 4.6), and l and u represent lower and upper bounds on the time between these events, where $l \in \mathcal{R} \cup \{-\infty\}$, $u \in \mathcal{R} \cup \{\infty\}$ such that $l \leq t(ev_2) - t(ev_1) \leq u$.

Qualitative Control Plan

The interaction of constraints explicitly specified in the QSP, with constraints due to plant dynamics, makes determination of feasible trajectories computationally intensive. Because the system must run in real time, we seek to minimize the dispatcher's runtime computation by pre-computing sets of feasible trajectories that satisfy both types of constraints. The plan compiler performs this off-line computation, as shown in Fig. 2, outputting a QCP that contains the feasible trajectories.

A QCP augments the input QSP by adding flow tubes and a dispatchable graph [Muscuttola, 1998]. The flow tubes represent feasible trajectory sets. The dispatchable graph represents temporal constraints in a form easily interpreted by the dispatcher. In this case, the temporal constraints represented in the QCP include both the temporal constraints explicitly specified in the QSP, and ones due to plant dynamics.

Definition 5 (QCP): A qualitative control plan (QCP) is a tuple $\langle q, F, g \rangle$, where q is the associated QSP, F is a set of flow tubes (Def. 6), and g is a dispatchable graph.

A flow tube is associated with a QSP activity. It represents feasible trajectories that result in successful execution of the activity. Thus, it is a function of the activity's constraints, and the dynamics of the activity's SISO system.

Definition 6 (Flow Tube): A flow tube is a set of trajectories, $Y = TUBE(a)$, defined over a time interval $[t_0, t_g]$, where a is a QSP activity, such that the goal and operating constraints of a are satisfied. Thus, a trajectory, $y(t) \in Y$ iff $\langle y(t_g), \dot{y}(t_g) \rangle \in R_{goal}(a)$ and all constraints in $R_{op}(a)$ are satisfied over the interval $[t_0, t_g]$, while obeying the dynamics of $S(a)$, as specified by Eq. (1).

An example flow tube is shown in Fig. 6a. A flow tube can be characterized as a set of cross-sectional regions in position-velocity phase space, one for each time in the interval $[t_0, t_g]$. Thus, such a cross section, r_{cs} , is a

function of the flow tube, and of time: $r_{cs} = CS(Y, t): t_0 \leq t \leq t_g$. Fig. 6b depicts cross sections for times t_0 , t_1 , and t_g .

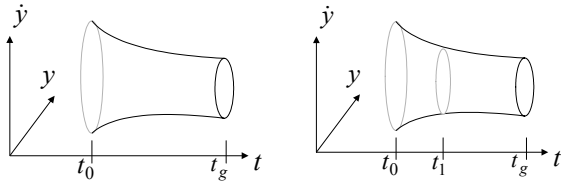


Fig. 6 – a. Example flow tube over interval $[t_0, t_g]$; b. cross sections at t_0 , t_1 , and t_g .

Next, consider the set, R_{cs} , of all cross sections in an interval $[t_0, t_1]$, where $t_0 \leq t_1 \leq t_g$. We use this set to investigate conditions under which the associated activity can be executed successfully for any start time in the interval $[t_0, t_1]$.

Theorem 1 (Temporal controllability of an activity): Let R_{cs} be a set of cross sections of a flow tube, Y , for activity a , where

$$R_{cs} = \bigcup_{t_0 \leq t \leq t_1} CS(Y, t)$$

If an allowed initial region, r_1 , is a subset of every cross section in R_{cs} ($r_1 \subseteq r_{cs} \forall r_{cs} \in R_{cs}$), then the duration of a is controllable over the interval $[l, u]$, where $l = t_g - t_1$, and $u = t_g - t_0$. Conversely, each cross section of Y of which r_1 is a subset corresponds to a controllable duration of a . Furthermore, if the position trajectories in Y change monotonically, then the set of controllable durations specified in this way by r_1 will form a contiguous interval.

The monotonicity assumption is crucial in that it avoids disjunctions in the temporal constraints.

We now extend temporal controllability concepts to sequences of activities. Recall that activity sequences, such as the CM1-CM2 sequence in Fig. 5, can be used to represent discontinuous changes in operating constraints. The transition from CM1 to CM2 represents a transition between qualitative states; from double to single support.

In order to ensure that a flow tube for CM1 does not contain “dead end” trajectories, we require that all trajectories in such a flow tube have a feasible continuation in a flow tube for CM2. Hence, we require that the goal cross section of a flow tube, Y_1 , for CM1 be a subset of a cross section of a flow tube, Y_2 , for CM2, as shown in Fig. 6a. Thus, if Y_1 is defined over the interval $[t_0(Y_1), t_g(Y_1)]$, and Y_2 is defined over the interval $[t_0(Y_2), t_g(Y_2)]$, then the following must hold: $CS(Y_1, t_g(Y_1)) \subseteq CS(Y_2, t_2)$, where $t_0(Y_2) \leq t_2 \leq t_g(Y_2)$. For example, t_2 may be $t_0(Y_2)$, corresponding to the initial cross section of Y_2 , as shown in Fig. 7a.

Suppose there exists a region, r_1 , which, along with Y_1 , determines a controllable interval $[l_1, u_1]$ for CM1, as discussed in Theorem 1. Suppose, also, that the goal region of Y_1 is a subset of the initial cross section of Y_2 , as

shown in Fig. 6a. Then, CM1 is temporally controllable over the range $[l_1, u_1]$, but CM2 must have a duration exactly equal to $t_g(Y_2) - t_0(Y_2)$. We now investigate ways to extend the controllable duration of CM2.

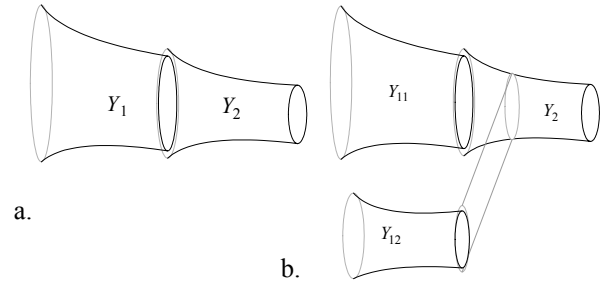


Fig. 7 – a. The goal region of flow tube Y_1 is a subset of a cross section of flow tube Y_2 . b. Two flow tubes that intersect with Y_2 .

Suppose there exists a set of flow tubes, Y_{1j} , for CM1 that all have the property that their goal regions are subsets of a cross section of Y_2 . Two such flow tubes are depicted in Fig. 7b. Suppose, further, that these cross sections of Y_2 are contiguous, and correspond to an interval $[l_2, u_2]$, where $u_2 = t_g(Y_2) - t_0(Y_2)$, $l_2 = t_g(Y_2) - t_2$, and $t_0(Y_2) \leq t_2 \leq t_g(Y_2)$.

If a region, r_1 , determines a controllable temporal range, $[l_{1j}, u_{1j}]$ for each Y_{1j} , and if the intersection of these temporal ranges is $[l_1, u_1]$, then CM1 is temporally controllable in the range $[l_1, u_1]$, and CM2 is temporally controllable in the range $[l_2, u_2]$. This concept can be applied recursively to successive activities in a sequence. In this way, if the initial state of the system is in r_1 , the controllable duration of all activities in the sequence are known. These controllable durations are the temporal constraints due to the plant dynamics. They are added to the temporal constraints specified explicitly in the QSP.

Plan Compiler

Plan compilation is accomplished in two steps. First, the dispatchable graph is computed based on the temporal constraints in the QSP. This graph represents the tightest temporal constraints on all activities. Second, flow tubes are computed for each activity, based on the temporal constraints for the activity specified in the dispatchable graph. The computation of the dispatchable graph is based on the Floyd-Warshall all pairs shortest path algorithm. This computation has been described previously [Muscuttola, 1998], hence we focus our discussion on computation of the flow tubes.

Flow tubes have a complex geometry. Therefore, any tractable flow tube representation will be an approximation of the feasible set. In order to ensure that any trajectory chosen by the dispatcher leads to plan execution success, we require our flow tube representation to include only feasible trajectories; the representation may include a

subset of all feasible trajectories, but not a superset [Kurzanski and Varaiya, 1999].

Our flow tube approximation uses polyhedral cross sections at discrete time intervals [Vestal, 2001]. The time interval chosen matches the control increment of the dispatcher. Therefore, the dispatcher will always be able to access flow tube cross sections for exactly the correct time. Fig. 8 shows a flow tube cross-sectional region in position-velocity phase space, and its polyhedral approximation. Note that the approximation is a subset of the true region; the approximation does not include points in state-space that do not belong to feasible trajectories.

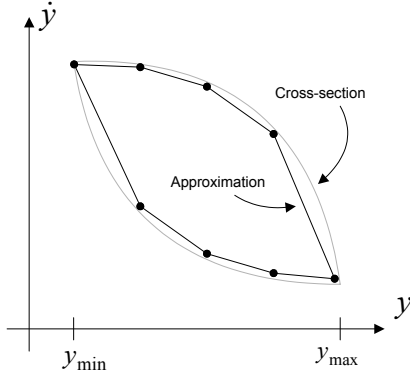


Fig. 8 – Flow tube cross section and approximation

In order to generate the polyhedral cross-sections, the plan compiler performs a *reachability analysis* that, for every vertex position, computes extreme corresponding velocities such that the resulting polygon contains only feasible trajectory points for the time associated with the cross section. We accomplish this reachability analysis by formulating constraints on cross section vertices as a linear program (LP).

The LP formulation is based on the analytical solution of Eq. (1). Eq. (1) is a 2nd-order linear differential equation, so its analytical solution is

$$\begin{aligned} y &= e^{\alpha t} (K_1 \cos \beta t + iK_2 \sin \beta t) + u / c \\ \dot{y} &= e^{\alpha t} (\beta(-K_1 \sin \beta t + iK_2 \cos \beta t) + \alpha(K_1 \cos \beta t + iK_2 \sin \beta t)) \end{aligned} \quad (2)$$

where

$$K_1 = y(0) - u / c, \quad K_2 = (i / \beta)(\alpha K_1 - \dot{y}(0))$$

$$\alpha = -k_d / 2, \quad \beta = \left(-i \sqrt{k_d^2 - 4k_p} \right) / 2, \quad u = k_p y_{set} + k_d \dot{y}_{set}$$

If we set the time, t , in Eq. (2) to a particular duration, d_i , corresponding to a particular cross section of interest, and if we fix gains k_p and k_d , then Eq. (2) can be expressed as

$$\begin{aligned} y &= f_1(y(0), \dot{y}(0), y_{set}, \dot{y}_{set}) \\ \dot{y} &= f_2(y(0), \dot{y}(0), y_{set}, \dot{y}_{set}) \end{aligned} \quad (3)$$

where f_1 and f_2 are linear for a particular setting of t , k_p , and k_d . Eq. (3) forms a set of equality constraints in the LP formulation. We also include a set of inequality constraints of the form

$$\begin{aligned} y_{set_min} &\leq y_{set} \leq y_{set_max} \\ \dot{y}_{set_min} &\leq \dot{y}_{set} \leq \dot{y}_{set_max} \end{aligned} \quad (4)$$

to represent the actuation limits, specified for the activity in the QSP. Further, we use a set of equality constraints to express

$$\langle y, \dot{y} \rangle \in R_{goal} \quad (5)$$

to ensure that state at the end of duration d_i is in the goal region, specified for the activity in the QSP. The formulation of (5) as a set of linear inequalities is straightforward because R_{goal} is required to be convex.

To compute a cross section for a particular R_{goal} and d_i , the plan compiler uses the formulation described by Eqs. (3 – 5), and sets the LP cost function to minimize $y(0)$. Solving this formulation yields the minimum initial position, y_{min} , shown in Fig. 8. Repeating this process with the cost function set to maximize $y(0)$ yields the maximum initial position, y_{max} . The compiler then establishes vertex positions at regular increments between y_{min} and y_{max} . For each such vertex position, the compiler solves the LP formulation with the cost function set to first minimize, and then maximize, $\dot{y}(0)$, in order to find the minimum and maximum velocities for that position. This results in a set of vertices in position-velocity state space, which form the polyhedral approximation, as shown in Fig. 8.

The compiler computes cross section approximations for every d_i in the temporal range $[l, u]$, where this range is given for each activity by the minimum dispatchable graph. This set of cross sections approximates a flow tube, such as the one shown in Fig. 6.

Consider, next, the problem of computing flow tube approximations for a sequence of activities, as in Fig. 7. As stated previously, the goal region for flow tube Y_1 in Fig. 7a must be a subset of a cross section of Y_2 . For the sake of completeness, we compute a separate set of cross sections for Y_1 for each cross section of Y_2 serving as a goal. This results in a set of flow tube approximations for Y_1 , as shown in Fig. 7b. Each flow tube approximation in this set represents valid trajectories that satisfy the plan goals.

Dispatcher

In order to execute a QCP, the dispatcher must successfully execute each activity in the QCP. The dispatcher accomplishes this by scheduling start and finish events, using the QCP's dispatchable graph, and by setting

control parameters for each activity such that the associated trajectory reaches the activity's goal region at an acceptable time.

In order to execute an activity, the dispatcher performs three key functions: initialization, monitoring, and transition. Initialization is performed at the start of an activity's execution, monitoring is performed continuously during the activity's execution, and transition is performed at the finish of the activity's execution.

For initialization, assuming that all trajectories begin in the flow tube of their activity, the dispatcher chooses a goal duration for the control activity that is consistent with its execution window [Muscuttola, 1998], and sets control parameters such that the state trajectory is predicted to be in the activity's goal region after the goal duration. The initialization function formulates a small *quadratic program* (QP) and solves it in order to determine these control parameters. This formulation is given in Fig. 9. Key to this formulation's simplicity is the fact that the analytic solution of Eq. 3 (functions f_1 and f_2) is used to predict the future state of the SISO system associated with the activity, and the fact that the formulation is guaranteed to produce a feasible solution, because the trajectory is within its flow tube. Further, presence of the trajectory in the flow tube guarantees that there exists a set of feasible control settings for all remaining activities in the plan, if there are no further trajectory disturbances.

FormulateControlQP($R_{goal}, y_{curr}, \dot{y}_{curr}, t_s, t_f$)

Parameters to optimize: $y_{pred}, \dot{y}_{pred}, y_{set}, \dot{y}_{set}$

Equality constraints: Eq. (3)

Inequality constraints: Eqs. (4, 5)

(Eq. (5) requires that trajectory prediction be within goal region)

Cost function

$$y_{goal} = (y_{\min}(R_{goal}) + y_{\max}(R_{goal})) / 2$$

$$\dot{y}_{goal} = (\dot{y}_{\min}(R_{goal}) + \dot{y}_{\max}(R_{goal})) / 2$$

$$Cost = (y_{goal} - y_{pred})^2 + (\dot{y}_{goal} - \dot{y}_{pred})^2$$

Fig. 9 - Dispatcher QP formulation.

After initializing an activity, the dispatcher begins monitoring execution of that activity. To monitor execution, the dispatcher continually checks whether the state trajectory remains in its flow tube. If this is not the case, then plan execution has failed, and the dispatcher aborts to a higher-level control authority. Such a control authority might issue a new plan in response to such an abort. For example, the biped trying to kick the soccer ball may give up on this goal if it is no longer feasible.

If the state trajectory is in its flow tube, the dispatcher checks whether it is on track to be in the goal region at the end of the goal duration. This check is accomplished by evaluating Eq. (3) for the current state, and checking whether the predicted state is within the goal region. If this is not the case, the dispatcher corrects this situation by adjusting control parameters using the QP formulation of

Fig. 9. Note that because the state trajectory is in its flow tube at this point, such a correction will always be possible.

As part of the monitoring function, the dispatcher also continually checks whether an activity's completion conditions are satisfied. Thus, it checks whether the state trajectory is in the activity's goal region, and whether the state trajectories of other activities whose completion must be synchronized are in their activity's goal regions. If all completion conditions for a control activity are satisfied, the dispatcher switches to the transition function.

If the activity being executed has a successor, the transition function invokes the initialization function for this successor. As part of this transition, the dispatcher notes the time of the transition event and propagates this through the temporal constraints using a local constraint propagation algorithm [Muscuttola, 1998]. This propagation tightens execution windows of future events. When all activities in the QCP have been executed successfully, execution terminates.

Results and Discussion

Fig. 10 shows an example sequence of flow tubes corresponding to lateral CM movement for activities CM1 – CM4 in Fig. 5. This sequence represents two steps, which takes approximately 1.4 seconds.

Fig. 11 shows initial cross sections for the flow tube set for activity CM1 such that the goal region for CM2 is achieved. The goal region is specified explicitly in the QSP. If the initial state of the biped is in the controllable initial region, then the goal region can be achieved after any duration in the range [0.7, 1.0]. Since CM1 and CM2 represent a single step (single support followed by double support), the temporal controllability for two steps is [1.4, 2.0]. Such controllability can be used to synchronize biped movement with that of a moving soccer ball in order to kick it, as shown in Fig. 12.

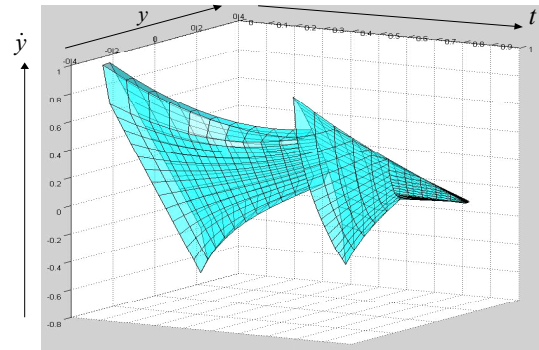


Fig. 10 – Example flow tube sequence.

If the initial state of the biped is outside the controllable initial region shown in Fig. 11, then temporal controllability is reduced. This expansion of the initial state region can be thought of as an enlargement of r_1 in Theorem 1, which results in r_1 being a subset of fewer cross sections. Thus, the requirement $r_1 \subseteq r_{cs} \forall r_{cs} \in R_{cs}$,

from Theorem 1 implies a smaller set R_{cs} , and a correspondingly smaller temporally controllable range. This trade-off is a direct result of the actuation limits and plant dynamics. Our plan compiler supports user control over this tradeoff through adjustment of required initial regions in the QSP.

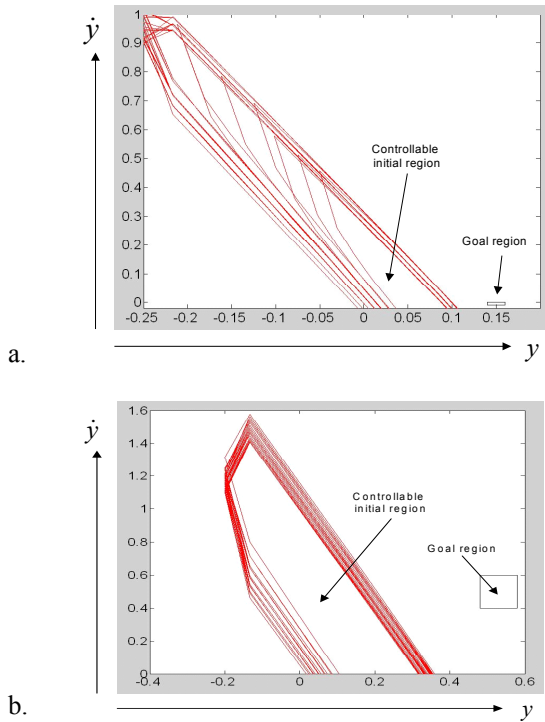


Fig. 11 – Initial cross sections for CM1 that achieve goal region in CM2; a. lateral, b. forward.

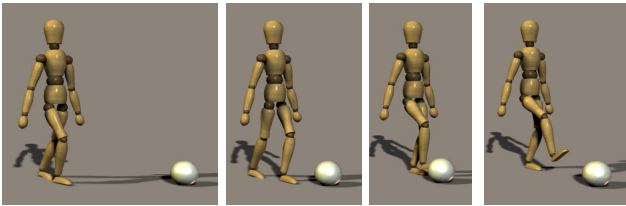


Fig. 12 – Walking to a moving soccer ball and kicking it.

An important challenge with this approach is the potentially large number of flow tubes in the set Y_{1j} that may have to be computed, especially when the method is applied recursively to longer activity sequences. This results in a fan-out for each predecessor activity. This problem can be mitigated by explicitly specifying tight temporal bounds in the QSP, and by explicitly specifying goal regions. An explicitly specified goal region serves as a root that terminates fan-out. An area of current research is incremental shifting of flow tubes. With such a capability, the full number of flow tubes in the set Y_{1j} would not have to be computed. Rather, a sparse subset would be computed, the elements of which would be shifted as needed.

References

- Bradley, E. and Zhao, F. 1993. *Phase-space control system design*. *Control Systems*, 13(2),39-46
- Frazzoli, E. 2001. *Robust Hybrid Control for Autonomous Vehicle Motion Planning*. Ph.D. Thesis, MIT
- Hofbauer, M. 1999. *Lyapunov Methods for Semiquantitative Simulation*. Ph.D. Thesis, TU Graz
- Hofmann, A., Massaquoi, S., Popovic, M., and Herr, H., 2004. *A sliding controller for bipedal balancing using integrated movement of contact and non-contact limbs*. *Proc. International Conference on Intelligent Robots and Systems (IROS)*. Sendai, Japan
- Kuipers, B., and Ramamoorthy, S. 2001. *Qualitative Modeling and Heterogeneous Control of Global System Behavior*. Hybrid Systems Control Conference.
- Kurzanski, A., and Varaiya, P. 1999. *Ellipsoidal Techniques for Reachability Analysis: Internal Approximation*
- Leaute, T., Williams, A. 2005. *Coordinating Agile Systems Through the Model-based Execution of Temporal Plans*. ICAPS, 2005
- Muscettola, N., Morris, P., and Tsamardinos, L. 1998. *Reformulating temporal plans for efficient execution*. *Proc. Of Sixth Int. Conf. On Principles of Knowledge Representation and Reasoning*
- Popovic, M., Hofmann, A., Herr, H. 2004. *Angular momentum regulation during human walking: biomechanics and control*. *Proceedings of the International Conference on Robotics and Automation (ICRA)*. New Orleans (LA, USA).
- Tsamardinos, I., Muscettola, N., Morris, P. 1998. *Fast Transformation of Temporal Plans for Efficient Execution*. AAAI
- Vestal, S. 2001. *A New Linear Hybrid Automata Reachability Procedure*. HSCC
- Vukobratovic, M. and Juricic, D. 1969. *Contribution to the Synthesis of biped Gait*. *IEEE Transactions on Bio-Medical Engineering*, Vol. BME-16, No. 1, 1969, pp. 1-6
- Williams, B. and Nayak, P. 1997. *A Reactive Planner for a Model-based Executive*. *Proceedings of the International Joint Conference on Artificial Intelligence*