Invariant Funnels for Underactuated Dynamic Walking Robots: New Phase Variable and Experimental Validation

Justin Z. Tang, A. Mounir Boudali and Ian R. Manchester

***Abstract*— We address the problem of finding useful invariant funnels for dynamic walking robots, i.e. sets of initial conditions from which continued walking in a stable manner is guaranteed. The construction is based on transverse dynamics and sum-of-squares verification. This paper makes two main contributions: firstly, we show that for typical models of walking robots the construction of such funnels can be significantly simplified by use of a new phase variable. Secondly, we provide the first hardware validation of the resulting funnels on an experimental testbed.**

I. Introduction

The development of legged robots that realise the stability, efficiency and agility of human walking has been the subject of intensive research in recent years (e.g. [1], [2], [3]). Underactuated "dynamic walkers" can demonstrate amazing feats of efficiency [4], but control design and stability analysis is inherently difficult since their dynamics are highly nonlinear, hybrid (mixing continuous dynamics with discrete impact events), and the target motion is a limit cycle (or more complex trajectory), rather than an equilibrium [3].

Efficient computation of basins of stability (or more generally forward-invariant sets) for dynamic walking robots would be an enabling technology for numerous practical problems. For example, they could be used to evaluate different robot designs [5], and to construct switching feedback controllers with guaranteed stability [6], [7].

The most well-known tool for analysing the stability of a nonlinear limit cycle is the Poincaré map [8], which describes the repeated passes of the system trajectory through a single transversal hypersurface. However, in general, the Poincaré map cannot be found explicitly for nonlinear systems and must be evaluated numerically. Furthermore, because the evolution of the system is analyzed only a single hypersurface, regions of stability in the full state space are difficult to evaluate, and design of continuous feedback control is not straightforward.

Basins of stability for walking robots have been evaluated using grid-based cell-to-cell mapping, e.g. [9], [10]. However, computational costs associated with exhaustive simulation grows exponentially with the dimension of the system. This motivates the search for alternative approaches.

In recent years, computational methods using sum-of-squares representations and semidefinite programming [11] have been developed for region-of-attraction estimation for equilibria, e.g. [12], [13], and constructing feedback controllers, e.g. [6], [14]. Notable applications have included analysis of the falling-leaf mode of an F/A-18 fighter [15] and control design for a perching glider [16].

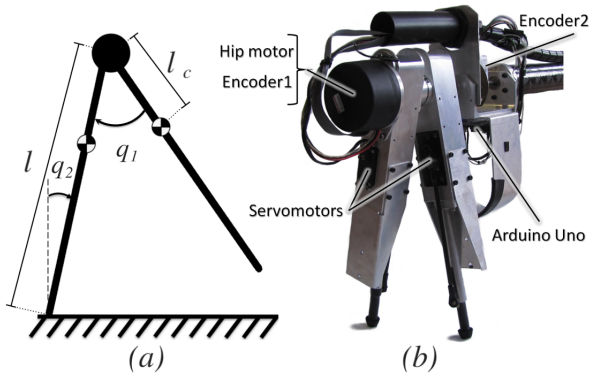

Fig. 1. (a) Schematic diagram of the classic Compass-gait dynamic walker, with actuation only at the hip; (b) Photo of the walker used in hardware verification.

These computational methods are extended to analyse the stability of nonlinear hybrid limit cycles in [17], [18] by leveraging the concept of transverse dynamics, previously applied to *local* stability analysis and control of walking robots in [19], [20]. In [17], [18] a new coordinate system is defined on a family of transversal hypersurfaces which move about the limit cycle in accordance to a phase variable. Stability certificates can then be to be computed on the new transversal coordinates.

In this paper, we make two significant contributions. Firstly, we demonstrate that for typical models of walking robots a significant simplification of the construction in [17], [18] is possible by use of a new phase variable. The relationship between this new phase variable and standard choice used in the *virtual constraints* methodology (e.g. in [2], [19]) is discussed in detail. Secondly, we validate for the first time the resulting controller and invariant sets on an experimental testbed, modelled after the classic Compass-gait dynamic walker as shown in Fig 1.

II. Problem Formulation and Preliminaries

Consider a hybrid system, with state space $x \in \mathbf{R}^n$ and continuous dynamics represented by

$$\dot{x} = f(x, u). \tag{1}$$

This work was supported in part by the Australian Research Council.

The authors are with the Australian Centre for Field Robotics, School of Aerospace, Mechanical and Mechatronics Engineering, The University of Sydney {j.tang, m.boudali, i.manchester}@acfr.usyd.edu.au

On a given switching surface $S^- \in \mathbf{R}^n$ the system (1) undergoes an instantaneous update

$$x^+ = \Delta(x^-) \qquad x^- \in S^-. \tag{2}$$

Suppose $x^\star(\cdot)$ represents a periodic walking motion for the system, i.e., a non-trivial T-periodic trajectory satisfying (1), (2), with $u^\star(\cdot)$ the associated input trajectory.

The overall objective is to compute an invariant funnel for periodic walking motion. That is, we seek to compute a region of state space $D \subset \mathbf{R}^n$ around $x^\star$, from which all solutions would remain in that region while maintaining forward motion. For the hardware in Fig 1, forward motion is defined as positive angular velocity for q_2, thereby inducing forward movement of the hip.

Our approach is to first construct a set of transverse dynamics in regions around $x^\star(\cdot)$ as suggested in [17], which in turn enables the search for a Lyapunov function in the transverse dynamics to prove the invariance condition. For completeness, we now briefly restate the transverse coordinate construction from [17], highlighting improvements and simplifications made in this paper.

We define a smooth local change of coordinates $x \mapsto (\tau, x_\perp)$. At each point $t \in [0, T]$, we define a hyperplane $S(t)$, with $S(0) = S(T)$. These transversal surfaces are defined by

$$S(\tau) = \{y \in \mathbf{R}^n : z(\tau)^T (y - x^\star(\tau)) = 0\} \tag{3}$$

where $z : [0, T] \to \mathbf{R}^n$ is a vector function defining the normal vector of each surface and would be optimised in Section III. $S(\tau)$ is a valid transversal surface iff $z(\tau)^T f(x^\star(\tau)) > 0, \forall \tau \in [0, \tau]$.

Given a point x nearby $x^\star(\cdot)$, the phase variable $\tau \in [0, T)$ represents which of these transversal surfaces $S(\tau)$ the current state x inhabits; the vector $x_\perp \in \mathbf{R}^{n-1}$ is the transversal state representing the location of x within the hyperplane $S(\tau)$, with $x_\perp = 0$ implying that $x = x^\star(\tau)$. More precisely,

$$x_\perp = \Pi(\tau)(x - x^\star(\tau)) \tag{4}$$

where $\Pi(\tau)$ is a projection operator constructed from $z(\tau)$, as will be discussed in Section IV.

The dynamics of the system in these new coordinates can be expressed as [17, Theorem 1] :

$$\dot{\tau} = \frac{z(\tau)^T f(x^\star(\tau) + \Pi(\tau)^T x_\perp)}{z(\tau)^T f(x^\star(\tau)) - \frac{\partial z(\tau)}{\partial \tau}^T \Pi(\tau)^T x_\perp} =: \frac{n(x_\perp, \tau)}{d(x_\perp, \tau)} \tag{5}$$

$$\dot{x}_\perp = \dot{\tau} \left[\frac{d}{d\tau}\Pi(\tau)\right] \Pi(\tau)^T x_\perp + \Pi(\tau) f\Big(x^*(\tau) + \Pi(\tau)^T x_\perp\Big) - \Pi(\tau) f\left(x^\star(\tau)\right) \dot{\tau}. \tag{6}$$

The construction of $z(\tau)$ and $\Pi(\tau)$ in this paper has been significantly simplified compared to that proposed in [17]. We achieve this by making three assumptions about the properties of the system in consideration: its dynamics; its switching surface S^-; and its target trajectory $x^\star$.

First, we assume that the state space can be represented in terms of the configuration space and its velocities $x = \left[q^T, \dot{q}^T\right]^T$; and that its dynamics can be written in the form $f = \left[\dot{q}^T, \hat{f}(q, \dot{q}, u)^T\right]^T$.

This assumption is satisfied for mechanical systems written in Euler-Lagrange form:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = B(q)u. \tag{7}$$

or equivalently represented in n-dimensional state space with $x = [q_1, q_2, \dot{q}_1, \dot{q}_2]^T$ and,

$$\hat{f}(q, \dot{q}, u) = M^{-1}(q)[-C(q, \dot{q})\dot{q} - G(q) + B(q)u]$$

Second, we assume that the switching surface S^- is a hyperplane which can be entirely represented in terms of the configuration space q. That is, it can be defined as $a^T q + b = 0$ for some vector a of length $n/2$ and some scalar b.

Third, we assume that the periodic walking motion $x^\star(\cdot)$ can be represented by a monotonic phase variable dependent only on q. In particular for our system, this can be guaranteed if $x^\star(\cdot)$ was designed by a set of virtual constraints. In a virtual constraint, a monotonic phase variable $\tau(q)$ is used to parameterize the motion of the robot within a step. Traditionally, $\tau(q)$ would be synthesized from angular measurements, an inclinometer, or some combination of these. We represent the virtual constraint like so:

$$q := \phi(\tau) = [\phi_1(\tau), \cdots, \phi_{n/2}(\tau)]^T. \tag{8}$$

III. Virtual Constraints and Phase Variable Selection

In this section, we first illustrate the use of a traditional choice of phase variable in a virtual constraint, and its incompatibility with regional analysis. We then provide a simplified and novel construction of a phase variable which addresses these issues and significantly simplifies the construction of transverse dynamics in Eqs. (5)-(6).

Throughout the remainder of this paper, we will illustrate our method with our hardware platform, modelled after the classic underactuated compass gait walker as shown in Figure 1. Here, q_1 is referred to as the 'swing leg' while q_2 is referred to as the 'stance leg'. The hybrid dynamics of the walker can be found in the Appendix. Note that we adopt the convention of positive angle as clockwise, hence the stance leg angle, θ_{st} is monotonically *increasing* through a footstep.

Foot impact occurs when the swing leg hits the ground. Hence, the switching surface S^- as defined in the discrete update Eq. (2) is

$$S^- = \{x \in \mathbf{R}^n : q_1 = 2q_2\}. \tag{9}$$

With appropriate parameter configurations and control law, the compass gait walker can follow a designated limit cycle trajectory. Figure 2 plots the 4-dimensional state space of one such limit cycle as tested in the hardware, with the stance leg dynamics superimposed with the swing leg dynamics. The green circles show the starting points for both the stance leg and the swing leg immediately after impact; while the magenta crosses illustrate the impact point.

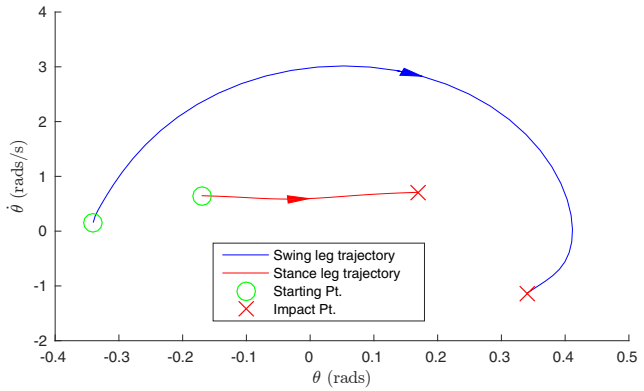

Fig. 2. The phase portrait for the compass gait hardware in limit cycle motion.

A. The Traditional Phase Variable on the Compass Gait

Typically in virtual constraint design for the compass gait walker [2], it is common to select the stance leg angle, θ_{st}, as the phase variable, i.e.,

$$\tau(q) = \theta_{st}.$$

In Figure 3, we highlight the use of this traditional choice of phase variable and the difficulties this presents in transverse stability analysis. The black solid line shows the same trajectory represented in Figure 2, with the black arrow indicating the direction of the dynamics. Red arrows drawn throughout the trajectory indicate the vector $z(\tau)$ which represents the direction in which the phase variable advances at that point.

In this case, the phase variable only advances in the direction of the stance leg angle throughout the cycle and hence θ_{st} is the only component in these vectors, i.e., $z = [0, 1, 0, 0]^T$. Note that monotonicity of the phase variable is equivalent to the red arrows forming acute angles with the direction of the black trajectory or $z^T \dot{x} > 0$ – i.e., the flow of the trajectory is constantly advancing the phase variable.

As per Eq. (3), the vertical blue lines in Figure 3 are surfaces transverse to $z(\tau)$. These transversal surfaces represent the set of states that can be associated with a particular phase value. For example, the leftmost transversal surface here represents the set of possible states that can be associated with $\tau \approx -0.17$ rads.

For simplicity, it is often assumed that switching occurs at a predesignated phase θ_{st}^d (see, e.g., [2, pp. 165-166]). This assumes the switching surface S^- to take the form,

$$S^- = \{x \in \mathbf{R^n} : \theta_{st} = \theta_{st}^d\}. \tag{10}$$

In Figure 3, this switching surface is equivalent to the leftmost transversal surface. However, this could only occur if one of the following two conditions hold true.

Condition 1: The point of impact always precisely equals that of on the nominal trajectory. This essentially assumes a finite-time controller guarantees the walker reach and remain on the target trajectory within one foot step.

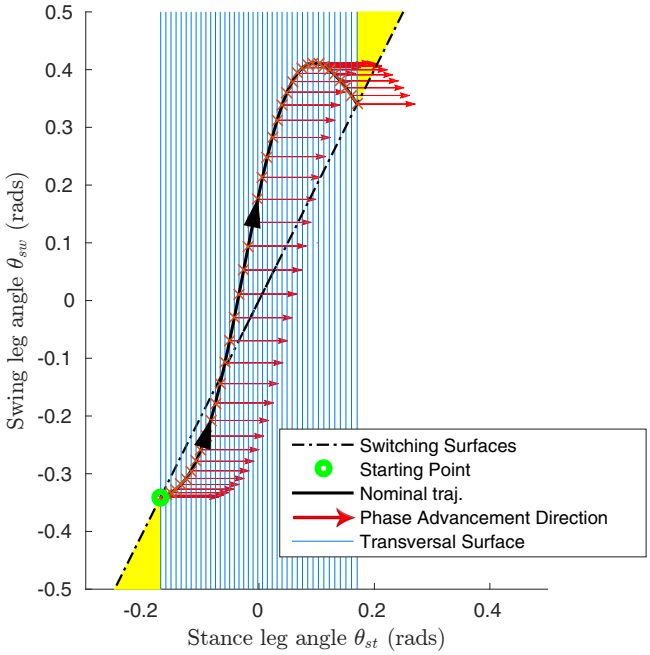

Fig. 3. Illustration of traditional phase variable for the compass gait walker – θ_{st}. Red arrows indicates the direction of $z(\tau)$ i.e., the direction in which the phase variable is increasing; blue lines indicate the transversal surfaces – the set of possible states corresponding to a particular phase value.

Condition 2: The walker has the ability to artificially trigger impact at the instant the designated final phase variable (θ_{st}^d) is reached.

Both Conditions 1 and 2 are arguably difficult to realise precisely with physical hardware. Indeed, if either conditions were not perfectly followed, impact would naturally occur when Equation (9) is satisfied, which is illustrated as black dotted line in Figure 3.

Given the discrepancy between the switching surface that this traditional choice of phase variable assumes – Eq. (10) – and the "natural interpretation" of the switching surface – Eq. (9) – the yellow shaded areas in Figure 3 mark problematic regions for analysis. In these regions – which occur when the Conditions 1 and 2 are not strictly met and the walker deviates from the nominal trajectory – the state would be undefined by this traditional phase variable. Therefore, any feedback controller expressed as a function of this phase variable would "run out of tape" in these shaded regions.

For transverse analysis, the shaded regions are undefined as they are not associated with any phase variable. Intuitively, if the leftmost and rightmost blue line aligned with the dotted line, the yellow shaded regions would disappear. Figure 4 shows the result of such arrangement, which will now be discussed.

B. Construction of a Novel Phase Variable

We now propose the construction of a new phase variable which aligns with the switching surface, overcoming the issues described in the previous section, and yet is significantly simpler than the construction proposed in [17].

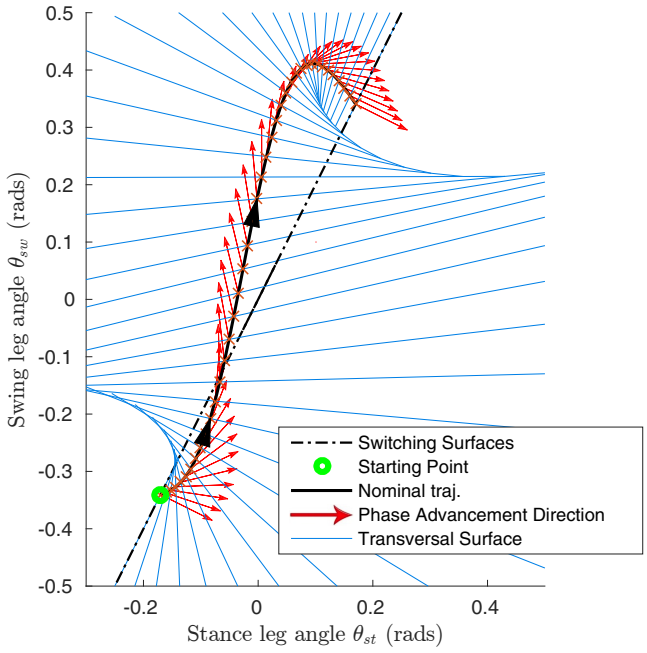

Fig. 4. Illustration of a new phase variable for the compass gait walker - a combination of both stance and swing angle. Red arrows indicates the direction $z(\tau)$, i.e. the direction in which the phase variable is increasing.

Geometrically from Figure 4, to align the transversal surfaces with the switching surface in Eq. (9) is equivalent to enforcing the direction of the two outermost red arrows being $[-1, 2, 0, 0]^T$. To simplify derivations of the transverse dynamics, we enforce $|z| = 1$, hence this condition becomes:

$$z(x^+) = z(x^-) = \left[\frac{-1}{\sqrt{5}}, \frac{2}{\sqrt{5}}, 0, 0\right]^T. \tag{11}$$

For the compass-gait walker, we propose a simple parametrization of z in terms of the angle of the red arrows in Fig 4, ψ:

$$z(\psi(\tau)) = [\cos(\psi(\tau)), \sin(\psi(\tau)), 0, 0]^T, \tag{12}$$

with

$$\frac{\partial z(\tau)}{\partial \tau} = [-\sin(\psi)\dot{\psi}, \cos(\psi)\dot{\psi}, 0, 0]^T.$$

Computations can be further simplified by parametrizing $\psi(\tau)$ as a Bezier polynomial, which allows the enforcement of (11) by simply fixing the first and last coefficients.

In addition to aligning with the switching surface, $z(\tau)$ must ensure that the transformation to transverse dynamics is well-posed. From Equation (5), it is apparent that the transformation $x \mapsto (x_\perp, \tau)$ is ill-defined when the denominator of $\dot{\tau}$, or $d(x_\perp, \tau)$, is zero, i.e.,

$$z(\tau)^T f(x^\star(\tau), u^\star(\tau)) - \frac{\partial z(\tau)}{\partial \tau}^T \Pi(\tau)^T x_\perp = 0. \tag{13}$$

As derived in [17], the smallest transversal $x_\perp^\dagger$ for which (13) is true has norm

$$|x_\perp^\dagger(\tau)| = \frac{|z(\tau)^T f(x^\star(\tau), u^\star(\tau)))|}{|\frac{\partial z(\tau)}{\partial \tau}|}.$$

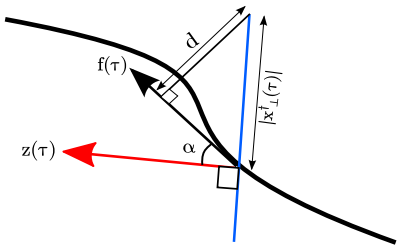

Fig. 5. Illustration of the distance, d, to be optimized in order to maximise the region that can be represented by the transverse dynamics.

In this paper, we modify the method of [17] by considering the distance from $x_\perp^\dagger$ to its projection on to the line spanned by $f(x^\star(\tau), u^\star)$, as illustrated in Fig. 5. The resulting distance d approximates the distance from $x_\perp^\dagger$ to the trajectory, and is computed like so:

$$d = |x_\perp^\dagger(\tau)| \cos(\alpha) = \frac{|z(\tau)^T f(x^\star(\tau), u^\star(\tau))|}{|\frac{\partial z(\tau)}{\partial \tau}|} z(\tau)^T \bar{f}(\tau), \tag{14}$$

where $\bar{f}(\tau) = \frac{f(x^\star(\tau), u^\star(\tau))}{|f(x^\star(\tau), u^\star(\tau))|}$ and α is the angle between $z(\tau)$ and $f(x^\star(\tau), u^\star(\tau))$.

However, note that the denominator in (14) contains $\frac{\partial z(\tau)}{\partial \tau}$. The optimum solution may contain constant $z(\tau)$ for some interval of τ; thereby causing (14) to go to infinity. Hence, instead of maximising (14), we find it is better numerically posed to optimise for z by minimizing the inverse of (14):

$$\arg\min_{z(\tau)} \left(\int_0^T \frac{|\frac{\partial z(\tau)}{\partial \tau}|^p}{|z(\tau)^T f(x^\star(\tau), u^\star(\tau)) z^T \bar{f}|^p} d\tau \right)^{1/p} \tag{15}$$

$$s.t. \quad z(x^+) = z(x^-) = \left[\frac{-1}{\sqrt{5}}, \frac{2}{\sqrt{5}}, 0, 0\right]^T$$
$$z(\tau)^T f(x^\star(\tau), u^\star(\tau)) > \delta.$$

The decision variables for the optimization are the coefficients of the Bezier polynomial defining ψ in (12). The authors have had success setting $p = 100$ and $\delta = 0.06$. With a desktop computer equipped with a 3.4 GHz Intel i7 and 24 GB of RAM, this can be solved within 5 seconds with ψ parameterised as a 5^{th} order Bezier polynomial, with results shown in Fig 4.

This construction is simpler than that in [17] since z only has non-zero entries in θ_{sw} and θ_{st}, the optimization for $z(\tau)$ can be directly computed from any virtual constraint definition in the form of (8).

IV. Transverse Analysis with a Geometric Phase Variable via Sums-of-Squares Programming

With the new simplified phase variable derived in Section III, we now demonstrate transverse stability analysis leading to an explicit region of stability in the full state space of the compass gait walker. Our approach proceeds as follows:

1) Compute the transverse dynamics using the simplified phase variable.

2) Design a stabilizing controller and derive an initial seed for the Lyapunov function.
3) Define τ-varying regions around $x^\star(\cdot)$ for which the invariance conditions will be checked.
4) Leveraging the S-procedure, iteratively maximise the invariance region using sum-of-squares optimization.

We now explore each of these four steps, highlighting improvements and simplifications made in this approach over that reported in [17].

A. Transverse Dynamics Computation

The new simplified phase variable, as defined by $z(\tau)$ in (12), enables significantly simpler computation for the transverse dynamics. For the compass gait, the projection operator, i.e. Π in (4), and its derivative can now be analytically computed:

$$\Pi(\tau) = \begin{bmatrix} -\sin(\psi(\tau)) & \cos(\psi(\tau)) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\frac{d}{d\tau}\Pi(\tau) = \begin{bmatrix} -\cos(\psi(\tau))\frac{d\psi}{d\tau} & -\sin(\psi(\tau))\frac{d\psi}{d\tau} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Note that the analytical construction of Π and $\frac{d}{d\tau}\Pi$ and the resulting sparsity of these variables significantly simplifies computations compared with methods suggested in [17]. Specifically, since $\left[\frac{d}{d\tau}\Pi(\tau)\right]\Pi(\tau)^T = 0$, the first term of (6) goes to zero, resulting in vastly simpler transverse dynamics:

$$\dot{x}_\perp = \Pi(\tau)(f_\perp(\tau) - f_\star(\tau)\dot{\tau}) \tag{16}$$

where $f_\perp(\tau) = f\Big(x^*(\tau) + \Pi(\tau)' x_\perp, u(x_\perp, \tau)\Big)$ and $f_\star(\tau) = f(x^\star(\tau), u^\star(\tau))$.

Using the transverse dynamics in (16), we can now construct Lyapunov functions which prove orbital stability by showing $x_\perp \to 0$ as $t \to \infty$.

B. Stabilizing Controller and Initial Seed for the Lyapunov Function

A natural candidate for a stabilizing controller and an initial seed of a Lyapunov function can be obtained via the solution of the transverse-jump-Riccati equation.

$$\begin{aligned} -\dot{P} &= A^T P + PA - PB_\perp R^{-1} B_\perp^T P + Q_\perp, & t \neq t_i \\ P(\tau_i^-) &= A_d(\tau_i)^T P(\tau_i^+) A_d(\tau_i) + Q_i, & t = t_i \end{aligned}$$

where $R, Q, Q_i > 0$;

$$B_\perp = \Pi(\tau)\frac{\partial f(x^\star(\tau), u^\star(\tau))}{\partial u} - \Pi(\tau) f(x^\star(\tau), u^\star(\tau))\frac{\partial \dot{\tau}}{\partial u};$$

and A is the linearization of $\dot{x}_\perp$ w.r.t. $x_\perp$. Given the simplification of the transverse dynamics in (16), a simplified expression of A over [17, Eq. (18)] is:

$$A(t) = \Pi(t)\left(\frac{\partial f(x^\star(t), u^\star(t))}{\partial x}\Pi(t)^T - f(x^\star(t), u^\star(t))\left.\frac{\partial \dot{\tau}}{\partial x_\perp}\right|_{x_\perp = 0}\right).$$

The solution of the transverse-jump-Riccati equation forms a locally stabilizing, phase-varying feedback controller:

$$u(\tau, x_\perp) = u^\star(\tau) - R^{-1} B_\perp(\tau)^T P(\tau) x_\perp. \tag{17}$$

The transverse-jump-Riccati solution also forms a locally valid Lyapunov function for the closed-loop system, $V(x_\perp, \tau)$ for a small region around $x^\star$ like so

$$V(x_\perp, \tau) = x_\perp^T P(\tau) x_\perp. \tag{18}$$

C. Invariant Funnels in the Full State Space

Using the transverse-LQR controller in (17), we verify the stability of the closed loop system by constructing invariant funnels around the nominal limit cycle. For brevity, we will hereafter refer to the closed loop system as follows:

$$f_{cl}(x) = f(x, u(\tau, x_\perp))$$

where $u(\tau, x_\perp)$ is defined in (17).

To construct the invariant funnels, we seek to compute a the τ-varying sets $D(\tau)$ such that for some $t_0 \in [0, \infty)$,

$$x_\perp(\tau(t_0)) \in D(\tau(t_0)) \implies x_\perp(\tau(t)) \in D(\tau(t)) \quad \forall t \in [t_0, \infty).$$

We describe this invariant funnel as a τ-varying sub-level set of a function $\check{V}(x_\perp, \tau)$:

$$D(\tau) = \{x_\perp \mid \check{V}(x_\perp, \tau) < \rho(\tau)\}. \tag{19}$$

In the continuous phase of the system, we require the invariance condition on the boundary of the funnel:

$$\check{V}(x_\perp, \tau) = \rho(\tau) \implies \dot{V}(x_\perp, \tau) < \dot{\rho}(\tau). \tag{20}$$

To allow the condition $\dot{\check{V}} < \dot{\rho}$ to be verified via sums-of-squares (SOS) programming, we multiply $\dot{\check{V}}$ through by $d(x_\perp, \tau)$, with $d(x_\perp, \tau)$ defined in Eq. (5). This results in an equivalent Lyapunov condition in (22), as shown on top of page 6. We also approximated the closed-loop dynamics with a third-order Taylor series expansion. We will discuss the ramification of the Taylor approximation in Section V.

In the discrete phase of the system, the condition to be verified at the time of switching is:

$$\check{V}(x_\perp^+, \tau^+) \leq \check{V}(x_\perp^-, \tau^-). \tag{21}$$

Applying the transverse transformation, this becomes:

$$\check{V}\left(\Pi(\tau_i^+)\Big[\Delta_i\left(x^\star(\tau_i) + \Pi(\tau_i^{-'})x_\perp\right) - x^\star(\tau_i^+)\Big], \tau_i^+\right) - \check{V}(x_\perp, \tau_i^-) \leq 0$$

$$DV := \frac{\partial \check{V}}{\partial \tau} n(x_\perp, \tau) + \frac{\partial \check{V}}{\partial x_\perp} \left\{ d(x_\perp, \tau)\Pi(\tau) f_{cl}\left(x^*(\tau) + \Pi(\tau)^T x_\perp\right) - \Pi(\tau) f_{cl}\left(x^\star(\tau)\right) n(x_\perp, \tau) \right\} \leq d(x_\perp, \tau)\dot{\rho} \quad (22)$$

D. Bilinear Search to Maximise Provably Stable Regions

Similar to [18], we verify the continuous (20) and discrete (21) invariance conditions on the boundary of the invariant funnel D by sampling $N = 40$ values of τ along the limit cycle trajectory.

In theory, we can parameterise the Lyapunov function $\check{V}$ by polynomial in $x_\perp$ and τ. However, this often leads to numerical problems in the resulting SOS program due to the high number of decision variables. We therefore parameterise $\check{V}$ by incorporating a τ-varying matrix Φ to the 'initial seed' of the Lyapunov function in (18), like so:

$$\check{V} = x_\perp^T (P(\tau) + \Phi(\tau)) x_\perp.$$

We can now maximise the invariant funnel for the limit cycle by using the integral of ρ as a surrogate for maximising the volume of the regions. Hence, the verification using SOS, with the S-procedure and Lagrange multipliers $l_1(x_\perp, \tau)$ and $l_2(x_\perp, \tau)$ is:

$$\max_{\check{V}, \rho(\tau), l_1, l_2} \int_0^T \rho(\tau) d\tau \quad (23)$$

subject to

$$-DV(x_\perp, \tau) - d(x_\perp, \tau)\dot{\rho}$$
$$-l_1(x_\perp)(\rho - V(x_\perp, \tau)) \in SoS, \quad (24)$$
$$d(x_\perp, \tau) - l_2(x_\perp)(\rho - V(x_\perp, \tau)) \in SoS, \quad (25)$$
$$\check{V}(x_\perp^-, \tau^-) - \check{V}(x_\perp^+, \tau^+) \in SoS, \quad (26)$$
$$V_{guess}(\Sigma_j e_j, \tau) - \check{V}(\Sigma_j e_j) > 0, \quad (27)$$
$$l_2 \in SoS. \quad (28)$$

Similar to [14], (27), is a constraint which prevents a large ρ to be returned simply by the scaling of Φ; where e_j is the j-th standard basis vector for the state space $\mathbf{R}^n$, and V_{guess} is defined by (18).

As can be seen, the above optimisation program is non-convex as the conditions are bilinear in the decision variables. Hence, we perform a bilinear search for the funnel as shown in Algorithm 1.

In our approach, we find that formulating ρ as a Chebyshev polynomial reduces numerical problems in the resulting SOS program, when compared with parameterizing it as a piecewise polynomial or with a monomial basis.

V. Numerical Results

Using the SPOTless toolbox [21] and commercial solver MOSEK version 7.1.0.46, we use Algorithm 1 to compute an invariant funnel for our hardware model. The funnel is shown in Fig. 6. We now address several observations from our numerical implementation.

Algorithm 1 Maximising Invariant Funnel Volume

1: Initialise: set $\Phi = 0$; and set ρ as the maximal constant (via bisection search) that satisfies (20) and (21) throughout the limit cycle trajectory;
2: converged = 0;
3: previousObj = 0;
4: **while** not converged **do**
5: **Multiplier-step**: Fix ρ and solve feasibility problem to find Lagrange multipliers l_1, l_2 satisfying (24) to (28).
6: ρ**-step**: Fix the l_1, l_2 and maximise the objective $\int \rho \, d\tau$ in (23) satisfying (24) to (28).
7: **if** $\frac{\int \rho \, d\tau - \text{previousObj}}{\text{previousObj}} < \epsilon$ **then**
8: converged = 1;
9: **end if**
10: previousObj = $\int \rho \, d\tau$
11: compare integral with previous; save current integral
12: **end while**

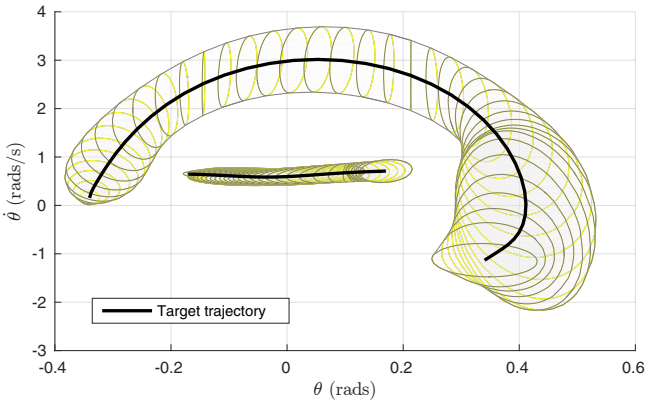

Fig. 6. Invariant funnel of the model based on the hardware.

A. Accuracy of Taylor Series Expansion

As mentioned in Section IV-C, we used a third-order Taylor approximation along the limit cycle to verify the invariance conditions using SOS. Hence, it was vital to ensure the conditions are also satisfied on the original Euler-Lagrange model.

For verification, we sampled 40,000 states on the boundary of the funnel with the Euler Lagrange model; and all achieved the required $\dot{V} < \dot{\rho}$.

B. Numerical performance and implementation details

Several strategies which the authors have utilised to overcome potential numerical issues common with large SOS problems are now discussed.

- **Numerical balancing**. Similar to [6], solutions to the jump-transverse-Riccati differential equation can have

a large range of eigenvalues, particularly for underactuated systems such as the compass gait. In order to avoid numerical issues in semidefinite programming, it is vital to find a coordinate transformation

$$x_b = \mathbf{T}x$$

where $\mathbf{T}$ numerically conditions the problem by ensuring the matrices $\mathbf{T}^T P \mathbf{T}$ from (19) and $\mathbf{T}^T(\mathbf{H}(DV))\mathbf{T}$ are as close to the identity matrix as possible; with $\mathbf{H}(DV)$ being the Hessian of DV from (22) evaluated at $x_\perp = 0$.

- **Solving a feasiblility problem to recover the analytic centre solution**. It is known that in practice, SDP solvers may not return strictly valid certificates for SOS programs, due to termination criteria and infeasible methods in these solvers, and fundamental limitations of floating-point implementations [22]. This poses a problem in the ρ-step of our method as the SDP solver often returns a maximised ρ that is slightly infeasible. The slightly questionable ρ in turn causes questionable solutions in the multiplier-step, thereby degenerating into worsening numerical problems in subsequent iterations. Solving the ρ-step problem twice – once as a *maximization* problem to optimize funnel volume; and a second time as a *feasibility* problem with a lower bound for the previously optimized objective – ensures a strictly feasible ρ is obtained at every iteration.

C. Example Implementation

A MATLAB implementation of the framework outlined herein has been made available online at [23].

VI. Hardware Experimental Verification

To test the efficacy of the proposed transverse-LQR controller and the verified invariant funnels in hardware, a compass gait walker was constructed as shown in Fig. 1.

Video footage of the hardware setup and experiments is included as a video attachment to this paper.

A. Hardware Experimental Setup and System Identification

The planar walker is mounted on a boom arm with a counterweight, and hence walks in a circular path. The dynamical effect of the boom and the counterweight is approximated by using a different hip mass for inertial (m_H) and gravitational (m_{H_g}) in the model. The full hybrid model is shown in the Appendix.

Optical encoders measuring the absolute angle for the inner leg, and the angle between the legs were installed, from which q_1 and q_2 could be calculated in real-time. An observer based on the linearized pendulum dynamics was designed to observe velocities $\dot{q}_1$ and $\dot{q}_2$.

Specific information regarding the construction (including CAD drawings) and system identification of the walker has been made available in [24].

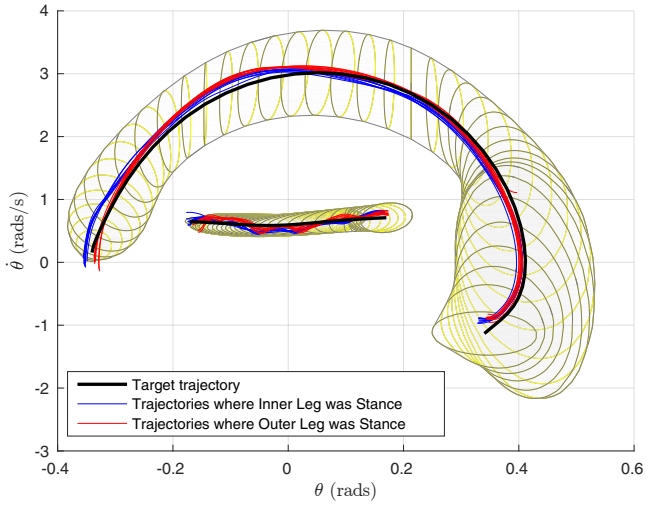

Fig. 7. Hardware experimental results superimposed with the computed invariant funnel.

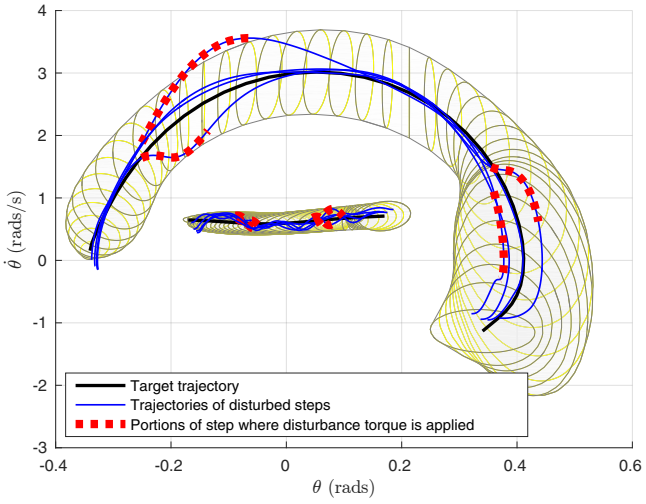

Fig. 8. Experimental results where disturbance torques were applied to the hip motor. Note that all trajectories either (a) successfully re-enters the invariant funnels after the disturbance; or (b) remain in the invariant funnel throughout the disturbance.

B. Hardware verification of computed invariance funnels

Fig. 7 shows the phase portrait of 15 steps for the walker while in stable walking motion. Note that while the vibrations from the boom could be observed in the stance leg; the trajectory remained within the invariant funnel as predicted. The only exception occurs at the very beginning of a few steps, which can be attributed to the vibrations during impact, unaccounted for in our impact model which assumes an inelastic collision with no slip or bounce.

To test the behaviour of the system closer to the boundary of the invariant funnel, a disturbance torque in the hip motor was applied at various stages of a step. The results are shown in Fig 8. Note that all trajectories successfully re-enter or remain in the invariant funnel after the disturbance torque has been applied.

VII. Conclusion

We have addressed the problem of finding useful invariant funnels for dynamic walking robots using transverse dynamics and sum-of-squares verification. Further, the first hardware validation of the resulting funnels were demonstrated on an experimental testbed.

While the compass gait walker has been used as illustration in this paper, this framework is applicable to higher dimensional systems and will be the focus of future work. Further, connections with robust stability analysis via transverse contraction [25] [26] will also be explored.

References

[1] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient bipedal robots based on passive-dynamic walkers," *Science*, vol. 27, no. 5712, p. 789, oct 2005.

[2] E. Westervelt, C. Chevallereau, B. Morris, J. Grizzle, and J. Ho Choi, *Feedback Control of Dynamic Bipedal Robot Locomotion*, ser. Automation and Control Engineering. CRC Press, jun 2007.

[3] J. W. Grizzle, C. Chevallereau, R. W. Sinnet, and A. D. Ames, "Models, feedback control, and open problems of 3D bipedal robotic walking," *Automatica*, vol. 50, no. 8, pp. 1955–1988, 2014.

[4] P. A. Bhounsule, J. Cortell, A. Grewal, B. Hendriksen, J. D. Karssen, C. Paul, and A. Ruina, "Low-bandwidth reflex-based control for lower power walking: 65 km on a single battery charge," *The International Journal of Robotics Research*, vol. 33, no. 10, pp. 1305–1321, 2014.

[5] L. Ning, L. Junfeng, and W. Tianshu, "The effects of parameter variation on the gaits of passive walking models: simulations and experiments," *Robotica*, vol. 27, no. 04, pp. 511–528, 2009.

[6] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, apr 2010.

[7] R. D. Gregg, A. K. Tilton, S. Candido, T. Bretl, and M. W. Spong, "Control and planning of 3-d dynamic walking with asymptotically stable gait primitives," *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1415–1423, 2012.

[8] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer-Verlag, 1997.

[9] A. Schwab and M. Wisse, "Basin of attraction of the simplest walking model," *ASME Design Engineering Technical Conference*, 2001.

[10] L. Liu, Y. Tian, and X. Huang, "A method to estimate the basin of attraction of the system with impulse effects: Application to the biped robots," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5314 LNAI, no. PART 1, pp. 953–962, 2008.

[11] P. A. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical Programming*, vol. 96, no. 2, pp. 293–320, may 2003.

[12] U. Topcu, A. Packard, and P. Seiler, "Local stability analysis using simulations and sum-of-squares programming," *Automatica*, vol. 44, no. 10, pp. 2669–2675, oct 2008.

[13] W. Tan, "Nonlinear Control Analysis and Synthesis using Sum-of-Squares Programming," Ph.D. dissertation, UC Berkeley, 2006.

[14] A. Majumdar, A. A. Ahmadi, and R. Tedrake, "Control design along trajectories with sums of squares programming," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4054–4061.

[15] A. Chakraborty, P. Seiler, and G. J. Balas, "Susceptibility of f/a-18 flight controllers to the falling-leaf mode: Nonlinear analysis," *Journal of guidance, control, and dynamics*, vol. 34, no. 1, pp. 73–85, 2011.

[16] J. Moore, R. Cory, and R. Tedrake, "Robust post-stall perching with a simple fixed-wing glider using LQR-Trees." *Bioinspiration & biomimetics*, vol. 9, no. 2, p. 025013, 2014.

[17] I. Manchester, "Transverse dynamics and regions of stability for nonlinear hybrid limit cycles," *Proceedings of the IFAC World Congress 2011 (extended version at arXiv:1010.2241)*, pp. 1–9, oct 2011.

[18] I. R. Manchester, M. M. Tobenkin, M. Levashov, and R. Tedrake, "Regions of Attraction for Hybrid Limit Cycles of Walking Robots," in *Proceedings of the IFAC World Congress 2011 (extended version at arXiv: 1010.2247)*, oct 2011.

[19] I. Manchester, U. Mettin, F. Iida, and R. Tedrake, "Stable dynamic walking over uneven terrain," *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 265–279, jan 2011.

[20] L. Freidovich, a. Shiriaev, and I. Manchester, "Stability analysis and control design for an underactuated walking robot via computation of a transverse linearization," *Proc. 17th IFAC World Congress, ...*, pp. 10 166–10 171, 2008.

[21] "SPOTless: Systems Polynomial Optimization Toolbox." [Online]. Available: https://github.com/spot-toolbox/spotless

[22] J. Lofberg, "Pre-and post-processing sum-of-squares programs in practice," *Automatic Control, IEEE Transactions on*, vol. 54, no. 5, pp. 1007–1011, 2009.

[23] "Simplified Construction for Stability Funnels for the Compass Gait Walker: Example MATLAB Code." [Online]. Available: http://www-personal.acfr.usyd.edu.au/ian/doku.php?id=wiki:software

[24] A. M. Boudali, F. H. Kong, J. Martinez, J. Z. Tang, and I. R. Manchester, "Design and Modeling of an Open Platform for Dynamic Walking Research," in *Proceedings of the IEEE International Conference on Mechatronics*, 2017.

[25] I. R. Manchester and J.-J. E. Slotine, "Transverse contraction criteria for existence, stability, and robustness of a limit cycle," *Systems & Control Letters*, vol. 63, pp. 32–38, sep 2014.

[26] J. Z. Tang and I. R. Manchester, "Transverse contraction criteria for stability of nonlinear hybrid limit cycles," *Proceedings of the IEEE Conference on Decision and Control*, no. February, pp. 31–36, 2014.

Appendix: Hybrid Dynamics of the Compass Gait Walker

The continuous dynamics of the compass gait walker, as shown in Fig. 1, can be represented in the form of Eq. (7), with $q = [\theta_{sw}, \theta_{st}]^T$, and

$$M(q) = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \tag{29}$$

$M_{11} = ml_c^2 + I_c; \quad M_{12} = M_{12} = -ml_c^2 - I_c + mll_c\cos(q_1)$
$M_{22} = 2I_c + \mathrm{m_H}l^2 + 2m(l^2 + l_c^2) - 2mll_c(1 + \cos(q_1))$

$$C(q, \dot{q}) = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \tag{30}$$

$C_{11} = 0; \quad C_{12} = -mll_c\dot{q}_2\sin(q_1)\,;$
$C_{21} = -mll_c\left(\dot{q}_1 - \dot{q}_2\right)\sin(q_1)\,; \quad C_{22} = mll_c\dot{q}_1\sin(q_1)$

$$G(q) = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} \tag{31}$$

$G_1 = \mathrm{g_0}ml_c\sin(q_1 - q_2)$
$G_2 = -\mathrm{g_0}\left(ml_c\sin(q_1 - q_2) + (2ml - ml_c + \mathrm{m_{Hg}}l)\sin(q_2)\right)$

$$B(q) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{32}$$

Assuming instantaneous and rigid impact of the swing leg without rebound and slip, the impact model is defined as:

$$\Delta_q = \begin{bmatrix} -1 & 0 \\ -1 & 1 \end{bmatrix} \tag{33}$$

$$\Delta_{\dot{q}}(q^-) = \Delta_q[H^+(q^-)]^{-1}H^-(q^-) \tag{34}$$

where $H^+_{1,1}(q^-) = p_1 + p_3$
$H^+_{2,1}(q^-) = H^+_{1,2}(q^-) = p_2\cos(q_1^-) - p_1 - p_3$
$H^+_{1,1}(q^-) = -2p_2\cos(q_1^-) + p_3 + 2p_1$
$H^-_{1,1}(q^-) = p_1 - p_2; \; H^-_{2,1}(q^-) = -p_1 + p_2$
$H^-_{1,2}(q^-) = p_3\cos(q_1) - p_1 + p_2; \quad H^-_{2,2}(q^-) = p_3\cos(q_1)$
$p_1 = m\,l_c^2 + I_c; \quad p_2 = lml_c; \quad p_3 = m_H l^2 + 2ml(l - l_c)$