Abstract—This paper presents the methodology used to achieve efficient and dynamic walking behaviors on the prototype humanoid robotics platform, DURUS. As a means of providing a hardware platform capable of these behaviors, the design of DURUS combines highly efficient electromechanical components with “control in the loop” design of the leg morphology. Utilizing the final design of DURUS, a formal framework for the generation of dynamic walking gaits which maximizes efficiency by exploiting the full body dynamics of the robot, including the interplay between the passive and active elements, is developed. The gaits generated through this methodology form the basis of the control implementation experimentally realized on DURUS; in particular, the trajectories generated through the formal framework yield a feed-forward control input which is modulated by feedback in the form of regulators that compensate for discrepancies between the model and physical system. The end result of the unified approach to control-informed mechanical design, formal gait design and regulator-based feedback control implementation is efficient and dynamic locomotion on the humanoid robot DURUS. In particular, DURUS was able to demonstrate dynamic locomotion at the DRC Finals Endurance Test, walking for just under five hours in a single day, traveling 3.9 km with a mean cost of transport of 1.61—the lowest reported cost of transport achieved on a bipedal humanoid robot.

I. INTRODUCTION

The humanoid robot, DURUS, was revealed to the public at the DARPA Robotics Challenge (DRC) Robot Endurance Test in June 2015 [1]. Developed by SRI International, DURUS was designed with the overarching goal of achieving never before seen efficiency in locomotion, thereby allowing for longer autonomous battery-powered operation. This goal is in response to the current state of the art in humanoid robots. While there have been dramatic increases in capabilities for performing tasks and navigating terrain through semi-autonomous task-based operation—as seen in the DRC Finals—this is often achieved at the cost of increased energy usage. The results presented in this paper take the opposite perspective by prioritizing a single objective: achieve maximum efficiency in locomotion through a holistic design, control and implementation methodology, with a focus on utilizing the full body dynamics of the robot—leveraging passive mechanical elements—to realize dynamic walking.

Traditional approaches for locomotion prioritize the ability to complete a wide-variety of tasks, e.g., step placement, turning, and stair climbing, over achieving highly dynamic and efficient locomotion. At the core of most methods employed on robots today is a low-dimensional representation of the full-order robot which utilizes a heuristic notion of stability known as the Zero Moment Point (ZMP) criterion [16], [25], and an extension termed Capture Point [20]. The ZMP and capture point methods are robust and allow for a variety of walking behaviors; however, the resulting locomotion is typically slow and very energy consuming. More recently, there have been several optimization based controllers [10], [17], [8], [9], [24] proposed in response to the DARPA Robotics Challenge. However, these approaches have been applied mainly with ZMP and capture point heuristics coupled with constraints, offer no formal guarantees, and again lack efficiency. With a view towards creating highly efficient walking, the passive dynamics community [18], [6], [4], [28] has aimed to utilize the passive dynamics of the robot to attain efficient walking with minimal power injection from actuators. While these walkers can achieve very efficient walking, e.g., the Cornell Ranger has the lowest recorded cost of transport for a legged robot of 0.19 [5], the design typically involves the use of many passive elements such as free-swinging joints and small actuators which make implementation difficult on a robot which must also have the ability to locomote and still perform a variety tasks.
This paper presents the methods used to demonstrate dynamic and efficient walking on the humanoid robot, DURUS, experimentally. This methodology begins with the design of dynamic and efficient walking gaits on bipedal robots through hybrid zero dynamics (HZD) [2], [12], [15], [26], [27], a mathematical framework that utilizes hybrid systems models coupled with nonlinear controllers that provably results in stable locomotion. In particular, we utilize HZD to formulate a nonlinear optimization problem for DURUS that accounts for the full-body dynamics of the robot in order to maximize the efficiency of the gait. The end result is a nonlinear controller that provably produces stable robotic walking [2]. The resulting trajectories are realized on the hardware via a feedforward term that encodes the formal gait design. To account for differences between the physical robot and the ideal model, a heuristic feedback is added to the control implementation in the form of regulators that modulate joints based upon environmental perturbations. The control framework presented allows for the full utilization of novel mechanical components on DURUS, including: efficient cycloidal gearboxes which allow for almost lossless transmission of power and compliant elements at the ankles for absorbing the impacts at foot-strike.

At the core of the control architecture implemented on DURUS is the underlying assumption that the dynamics of the electromechanical system will operate near those of the desired system. The low-level motor controllers of the robot ensure that the formally generated trajectories will be closely tracked, with 0.005 rad rms tracking error for the experiments documented in this paper. Also, due to the relatively small stabilizing perturbations induced by the feedback regulators, the walking trajectories demonstrated on DURUS in this work are shown to preserve 83.3% of the “formal” gait. Additionally, through the combination of formal controller design and novel mechanical design, the humanoid robot DURUS was able to achieve a mean electrical cost of transport of 1.61 over roughly five hours of continuous walking—the lowest recorded electrical cost of transport for a bipedal humanoid robot.

The presented work is structured as follows: The overview of the mechanical components, which provide an efficient basis for which the control strategy builds upon, is described in Sec. II. The mathematical modeling of DURUS is presented in Sec. III. The control approach, including the formal feedforward gait construction and the feedback regulator structure are detailed in Sec. IV. Finally, the experimental results are presented in Sec. V.

II. DESIGN

The underlying mechanical and electrical components incorporated into the design of DURUS provide an essential foundation from which the control design can build upon. In particular, a two-pronged approach was taken in the design of DURUS: (1) novel mechanical and electrical components providing significant gains in efficiency and (2) a leg morphology which was the result of an iterative feedback loop between mechanical design and control synthesis. The components resulting from this approach are shown in Fig. 3.

Novel Components. The primary mechanical components which provided gains in overall efficiency were the actuator and transmission elements (see Fig. 3). Each actuator-gearbox combination consisted of an electric motor connected via a chain reduction to a custom-designed cycloid transmission, which can achieve up to 97% efficiency. Each actuator-gearbox unit was lightweight, weighing only 2.7 kg and able to output 250 Nm of torque with maximum joint accelerations exceeding 130 rad/s².

To ultimately realize dynamic and efficient locomotion on the humanoid robot DURUS, precision in control implementation is required at every level of the hardware. Therefore, an essential component in the process of realizing locomotion is a motor controller which can accurately track the trajectories generated in Sec. IV accurately. Custom motor controllers are employed on DURUS, allowing for 10 kHz control of torque, current, and position. For the duration of the walking analyzed in Sec. V, these motor controllers tracked joint positions with an overall rms error of 0.005 rad and a peak error of 0.026 rad. Additionally, DURUS is self-powered with a 1.1 kWh battery pack weighing 9.5 kg.

Control in the Loop Design. The morphology of DURUS, and specifically the role of passive-compliant elements,
rectly impacted how well a control scheme could achieve efficient and stable dynamic gaits both in simulation and experimentally. The leg morphology of DURUS is the result of an iterative collaboration between the designer and control engineers. Specifically, designs for the leg geometry and passive-compliant ankles were passed to rigorous simulation for evaluation. In particular, the nonlinear control and gait approach in [14] was utilized to realize walking in simulation and the design was evaluated with regard to performance parameters such as the joint torques and walking stability. These findings were then compiled and passed back to the design engineer for improvement. The result of this iterative process was a leg design which walked in simulation with worst case torques of 150 Nm, as opposed to initial leg designs which demonstrated peak torques of 450 Nm; this procedure, along with several of the leg designs and their associated simulation torques, is illustrated in Fig. 4. The authors believe that this “control in the loop” mechanical design methodology was a key factor in the ability of the control scheme presented in Sec. IV to maintain smooth, stable walking while exploiting the energy saving capabilities of the passive-compliant ankle structures.

A key difference between DURUS and many humanoid robots is the use of passive springs in the ankles with significant compliance. To leverage the greatest returns, the springs are much more compliant than typically seen on powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveraging springs for efficiency [21], [30]. This level of compliance in DURUS allows for powered humanoid robots leveragi...
Hybrid Model. Having introduced the joint configuration of DURUS, the two-domain hybrid system model for DURUS can be defined based on the framework established in [23], [15]. In particular, we specify the robotic system as a tuple:

\[ \mathcal{H} = (\Gamma, \mathcal{D}, \mathcal{W}, S, \Delta, FG) \]

where \( \Gamma = (V, E) \) is a directed cycle specific to this walking model with vertices \( V = \{ ds, ss \} \) and edges \( E = \{ ds \rightarrow ss, ss \rightarrow ds \} \), \( \mathcal{D} = \{ \mathcal{D}_v \}_{v \in V} \) is a set of admissible domains, \( \mathcal{W} = \{ \mathcal{W}_v \}_{v \in V} \) is a set of admissible controls, \( S = \{ S_e \subset \mathcal{D}_v \}_{e \in E} \) is a set of guards or switching surfaces, \( \Delta = \{ \Delta_e \}_{e \in E} \) is a set of reset maps, and \( FG = \{ FG_v \}_{v \in V} \) is a set of affine control systems defined on \( \mathcal{D}_v \).

The two domains of the robot upon which the hybrid system is modeled depends on the current status of the foot contact points with the ground. The robot is in the double-support (DS) domain when both feet are in contact with the ground and transitions to the single-support (SS) domain when one of the legs lifts off the ground. The transition from single-support to double support domain occurs when the non-stance foot strikes the ground.

The continuous dynamics of the system depends on the Lagrangian of the robot model and the holonomic constraints, such as foot contacts with the ground, defined on a given domain. With the mass, inertia and length properties of each link of the robot, the equation of motion (EOM) for a given domain \( \mathcal{D}_v \) is determined by the classical Euler-Lagrange equation [19], [12]:

\[ D(q)\ddot{q} + H(q, \dot{q}) = B_u \dot{u} + J_v^T(q)F_v \]

where \( v \in V, D(q) \) is the inertia matrix including the reflected motor inertia, \( H(q, \dot{q}) = C(q, \dot{q})\dot{q} + G(q) + \kappa(q, \dot{q}) \) is a vector containing the Coriolis term, the gravity vector, and the spring force vector, \( B_u \) is the distribution matrix of actuators, and \( F_v \) is a collection of contact wrenches containing the external forces and/or moments exerted on the robot due to the holonomic constraints (see [19]). The contact wrenches are determined by enforcing holonomic constraints to be constant, i.e., the second order differentiation of the holonomic constraints is zero:

\[ J_v(q)\dddot{q} + J_v(q, \dot{q})\dot{q} = 0 \]

Combining both (2) and (3), results in the affine control system [23], \( \dot{x} = f_v(q, \dot{q}) + g_v(q)u_v \), where \( x = (q, \dot{q}) \in \mathbb{R}^{2n} \) are the states of the system.

IV. Control Design

To complement the efficient design of the robot, a control approach which leverages the passive dynamics of the robot while providing stability through the dynamic walking motions is introduced. First, a formal mathematical framework that utilizes hybrid system models coupled with nonlinear controllers and an optimization that provably results in the generation of stable walking trajectories is introduced. Implementation of these trajectories on hardware is realized through their time-based playback on embedded level position controllers, which we term the feedforward component of the control implementation. Finally, a heuristic feedback control structure is added to regulate and stabilize the robot about the formal walking trajectories and again implemented at the embedded level; this is referred to as the feedback control component. Specifically, joints are controlled at the embedded level through desired trajectories of the form:

\[ \hat{q}^d = q^d_0(t) + \Delta q^d(\tau, \phi_b) \]

where \( q^d_0 \) is the feedforward trajectory, \( \tau \) is a state based parameterization of time, \( \phi_b \) is the orientation of the body reference frame \( R_b \), and \( \Delta q^d(\tau, \phi_b) \) is the perturbation induced by the regulators.

A. Feedforward Control

A feedforward control scheme which generates a stable walking gait leveraging the passive-compliant elements in the system is presented in this section. The walking gait is the result of a nonlinear optimization problem and ultimately generates the time-based trajectories which compose the feedforward component of the control implementation.

Virtual Constraints. Analogous to holonomic constraints, virtual constraints are defined as a set of functions that regulate the motion of the robot with a certain desired behavior [2], [31], with the key difference that the virtual constraints are realized through control inputs, rather than contact wrenches. We start by defining the outputs on the robot that will be used in the next section to modulate the walking behavior. Inspired by [3], the linearized forward hip velocity, \( y_{hip}(q, \dot{q}) = \delta \dot{q}_{hip}(q, \dot{q}) \), is picked as the velocity-modulating output for both domains, where \( \delta \dot{q}_{hip}(q) \) is the linearized hip position of the robot.

Position-modulating outputs are chosen for each domain [2]. The number of outputs chosen per domain is determined by the number of available degrees of freedom in the respective domain. The double support domain, \( \mathcal{D}_{ds} \), is described by the nine outputs: stance knee pitch, stance torso pitch, stance ankle roll, stance torso roll, stance hip yaw, waist roll, waist pitch, waist yaw, and non-stance knee pitch. In single support, \( \mathcal{D}_{ss} \), the non-stance foot is no longer constrained.
to the ground so five additional outputs are introduced: the non-stance slope, non-stance leg roll, non-stance foot roll, non-stance foot pitch, and non-stance foot yaw.

With the definition of the outputs in hand, virtual constraints are defined as the difference between the actual and desired outputs of the robot [2], [26], [31]:

\[
y_{1,v}(q, \dot{q}, v_d) = y_{1,v}^d(q, \dot{q}) - v_d,
\]

\[
y_{2,v}(q, \alpha_v) = y_{2,v}^d(q) - y_{2,v}^d(t, \alpha_v),
\]

for \( v \in V \), where \( v_d \) is the desired hip velocity, \( y_{1,v} \) and \( y_{2,v} \) are relative degree 1 and (vector) relative degree 2 by definition, respectively. Moreover, a forth-order Bézier polynomial is adopted to specify each desired relative degree 2 output as in [27].

**Gait Generation.** With the goal of driving the outputs to zero exponentially, we consider the feedback linearizing controller formulated in [2]. The application of this control method yields linear outputs of the form:

\[
y_{1,v} = -\varepsilon y_{1,v},
\]

\[
y_{2,v} = -2\varepsilon y_{2,v} - \varepsilon^2 y_{2,v},
\]

for \( v \in V \) with \( V \) the set of vertices for \( \Gamma \) given in (1).

When the control objective is met such that \( y_{2,v} = 0 \) for all time then the system is said to be on the partial zero dynamics surface [2]:

\[
\text{PZ}_{v} = \{ x \in \mathbb{R}^{2a} : y_{2,v}(x, \alpha_v) = 0, L_f y_{2,v}(x, \alpha_v) = 0 \}.
\]

This surface will be rendered invariant through the use of the control law over the continuous dynamics of the system. However, it is not necessarily invariant through the discrete impacts which occur when the swing foot comes in contact with the switching surface. As a result, the parameters \( \alpha_v \) of the outputs must be chosen in a way which renders \( \text{PZ}_{v} \) invariant through impact, i.e., which yield partial hybrid zero dynamics (PHZD) and can be formulated as a constrained nonlinear optimization problem [2]:

\[
\alpha_v^* = \arg\min_{\alpha_v} \text{Cost}(\alpha_v) \quad \text{s.t.} \quad \Delta_v(S_c \cap \text{PZ}_{v}) \subseteq \text{PZ}_{v},
\]

where \( \Delta_v \) is the set of vertices for \( \Gamma \), \( S_c \) is the set of vertices for the partial zero dynamics surface, and \( \text{Cost}(\alpha_v) \) is the cost function.

To effectively implement the walking gait produced from the optimization problem on hardware, the desired joint and angular velocities of the robot in each iteration must be found. To obtain a set of time-based trajectories for playback on the physical hardware, DURUS is simulated using the feedback linearizing controller in [2] and the parameter set obtained from the optimization. The joint trajectories of the stable walking in simulation are recorded and stored as a set of time-based positions and velocities for tracking:

\[
q_{0}^d(t) = q_{\text{sim}}(ctf)
\]

\[
\dot{q}_{0}^d(t) = \ddot{q}_{\text{sim}}(ctf)
\]

where \( c_f \) is a scaling constant which is used to allow for the walking trajectories to be sped up or slowed down when implemented on hardware as the feedforward term in (4).

**B. Feedback Control.**

While the feedforward time-based trajectories, \( q_{0}^d \), are generated with the dynamics of the system in mind, it quickly became evident as gaits were implemented on DURUS that regulating feedback control would be crucial to stabilize the system for long-duration walking. The authors adopted a regulator design similar to those of [11], [22], but with a focus on position control. In particular, taking the actual and desired objectives for a given regulator, a trajectory perturbation was heuristically calculated and incorporated into the formally generated trajectories to yield a system which was responsive to minor destabilizations such as unmeasured compliance or environmental factors. The primary function of these trajectory perturbations is to smoothly stabilize the robot in the lateral (roll) direction and to steer the robot.

The feedback component of the controller is achieved through two regulators: a roll regulator for lateral stability and a yaw regulator for walking direction control. These regulators were implemented using discrete logic to handle a smooth blending factor dependent on the current discrete domain (\( D_{ss} \) or \( D_{ns} \)) and therefore prevents large jumps in the commanded position that can occur through the transitions between domains. The discrete logic of the blending factor is implemented using the global phase variable, \( \tau \), which is the normalization of the position of the hip from the beginning to end of the step. Through the course of each step, a normalized phase variable, \( \lambda \), is calculated as:

\[
\lambda = \frac{\tau - \tau_{\text{min}}}{\tau_{\text{max}} - \tau_{\text{min}}},
\]

where \( \tau_{\text{max}} \) and \( \tau_{\text{min}} \) are the maximum and minimum values of \( \tau \) as generated in the formal walking gait.

The implementation of the blending factor \( s \), and how it behaved with regard to the spring based switching, played a critical role in the behavior that the regulators induced. Throughout the single-support domain, \( D_{ss} \), the non-stance leg blending factor \( s_{\text{ns}} \) was increased according to \( \lambda \), starting from a magnitude of zero and finishing at one as the swing phase ends. The stance leg blending factor \( s_{\text{s}} \) is decreased at a rate faster than the duration of the whole domain using an acceleration factor \( c_f \). The single-support blending factor update can be summarized as:

\[
s_{\text{ns}} = \lambda, \quad s_{\text{s}} = -c_f \lambda,
\]

where \( s \leftarrow s + \Delta s \) in each cycle. During the double-support domain, \( D_{ss} \), the blending factor is held constant, such that each of the legs do not oppose the motion of the other. This blending factor feature will be used in the following sections to regulate the allowable trajectory perturbations in each of the discrete-mode feedback controllers.

**Roll Regulator.** The main stabilizing action of the roll regulator is to abduct the hip joint of the swing leg throughout the single support phase. This hip abduction effectively
changes the location of the foot strike, placing it in a more desirable configuration for recovering from a lateral sway. Also incorporated into the roll regulator is a waist roll action which moves the torso away from a position in which it might topple over the stance leg after impact. The feedback controller then takes on the form of a proportional controller:

\[ \Delta q^d = -s_i k \left( y^a - y^d \right), \quad i \in \{ s, ns \}, \]

where \( y^a \) are the actual torso orientation from the IMU and \( y^d \) are the desired torso orientation from the IMU and the torso orientation computed via the robot kinematics assuming a flat foot in contact with the ground.

**Yaw Regulator.** The yaw regulator uses user input via a joystick controller as the desired objective. While operating the robot on a treadmill, the electromagnetic interference was sufficient to render the heading information from the magnetometer unusable thus the actual heading is set to zero. This allows the user to modulate the steering action about the current direction that DURUS is facing. The desired effect of this regulator is to yaw the hip joint while the leg is in swing during single-support in order to change the orientation with which the foot will strike the ground. After double support, the hip yaw is blended away with the foot planted, turning DURUS about the stance leg and into the desired direction. The regulator is of the same proportional form as (15) where \( y^a := 0 \) and \( y^d := \) joystick input (see Fig. 7).

**C. Control Implementation.**

The control software infrastructure followed that of [7]. Specifically, each joint on the robot had a corresponding microcontroller communicating with a real-time (RT) process. This RT process relays data to and from a lower priority real-time process implemented in C++ to perform high-level control. The high-level control is relatively simple: the hardware coordinates are transformed into model coordinates via a transmission, then the pre-recorded trajectories played back with the regulators superimposed to yield the desired profiles of each joint as in (4). Finally, the model coordinates are transformed back into hardware coordinates.

From the high-level, desired joint trajectories travel through two stages before reaching the low-level embedded control. First, desired motor positions are sent from the high-level process to the Simulink-generated process responsible for communicating with the joint microcontrollers. At this stage, the trajectory is up-sampled from 250 Hz to 1 kHz with a first-order hold and sent to the embedded level. An IMU mounted in the torso of DURUS is used to determine the global orientation of the torso for the roll regulator. Finally, incremental encoders and absolute encoders are used at each joint, though incremental encoders are the only sensors used actively in the joint-level feedback control. The joint microcontrollers implement the modulated joint trajectories (4) and the joint velocities from (12) via position control at 10 kHz. The discrete modes of the regulator structure are triggered by the foot interactions with the ground; specifically, strike detection is triggered when the

Spring deflection in the ankles passes a certain threshold. The results of applying the regulators to DURUS while walking, including the discrete structure of the blending factor together with actual and desired values in (15) and the resulting applied delta, \( \Delta q^d \), is shown in Fig. 7.

**V. RESULTS**

The Robot Endurance Test at the DRC Finals took place over two days, during which DURUS exhibited sustained walking over large distances with a consistently low cost of transport [1]. These results were realized through the application of the formal techniques previously discussed coupled with a stabilizing regulator structure. The longest of these walking runs took place during the second day, during which DURUS walked on two consecutive batteries and exceeded walking distances of 3.8 km. A summary of the walking for this day is summarized in Table I.

Walking data was periodically recorded in ten minute segments while demonstrating the robot to the public. Live performance metrics, including the electrical cost of transport, were available on a screen displayed next to DURUS.

**TABLE I: Walking Statistics**

<table>
<thead>
<tr>
<th>Battery</th>
<th>Duration (hh:mm:ss)</th>
<th>Distance Traveled (meters)</th>
<th>( \bar{c}_{er} ) (mean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2:35:43</td>
<td>2055.0</td>
<td>1.57</td>
</tr>
<tr>
<td>2</td>
<td>2:17:12</td>
<td>1812.8</td>
<td>1.69</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>4:52:55</td>
<td>3867.8</td>
<td>1.61</td>
</tr>
</tbody>
</table>
The cost of transport data available for the duration of the experimental run on one battery charge is plotted in Fig. 8. The specific cost of electrical transport $c_{et}$ is calculated as in [7], where the total energy consumed over the weight and distance traveled is represented for step $i$ as:

$$c_{et,i} = \frac{1}{mgd_i} \int_{t_{i-1}}^{t_i} P_{el} dt,$$

where $P_{el}$ is the consumed power and $d_i$ is the x-position traveled by the non-stance foot of the robot through the duration of the $i$th step. Power data was computed directly from current and voltage measured on the battery pack which supplies power to all components on the robot.

In this work, energy efficiency is used as a metric in which to evaluate the mechanical design and control implementation. The reported electrical cost of transport for several robots is summarized in Table II, from which we observe the robots utilizing passive elements, small motors, or anthropomorphic designs to leverage energy savings demonstrating the lowest energy expenses (Cornell Ranger and Biped). Additionally, robots employing HZD to achieve locomotion exhibit efficient locomotion (AMBER 1 and 2D-DURUS), although these are restricted to walking in a 2D plane. The closest efficiency numbers come from ATRIAS—possibly since it inspired the compliment elements in the design of DURUS—yet this robot is not humanoid in nature. Therefore, in the category of full-scale bipedal humanoid robots (e.g., ATLAS and ASIMO) the electrical cost of transport on DURUS is the lowest ever reported.

A core contribution of this paper is the use of formal nonlinear control methods to realize dynamic walking on humanoid robots. To examine the degree to which the formal walking behavior is realized, we quantify the percentage of formal walking present in the gait experimentally through:

$$\%q_f = \frac{1}{n} \sum_{j=1}^{n} 1 - \frac{\Delta q_j^f(\tau, \theta)}{q_{0,j}^f(\tau, \alpha)},$$

for the $n = 17$ joints. The mean of 1588 steps over a 10 minute interval is shown in Fig. 10. Pictured are means of several of the regulated joints along with the mean percent formal walking of the total system. Specifically, the mean percent formal walking exhibited on DURUS was $\%q_f = 83.3\%$. Fig. 11 shows a visual comparison of the walking gait in simulation and experimentation over one step.

VI. CONCLUSIONS

This paper presented the methodology by which efficient locomotion was achieved on the humanoid DURUS. The electromechanical design of the robot was introduced, with a special focus on components that improved efficiency along

<table>
<thead>
<tr>
<th>Name</th>
<th>$\bar{c}_{et}$</th>
<th>$m$ (kg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>ATLAS [5]</td>
<td>5.3</td>
<td>102</td>
</tr>
<tr>
<td>ASIMO [6]</td>
<td>3.23</td>
<td>52</td>
</tr>
<tr>
<td>AMBER 1 [29]</td>
<td>1.88</td>
<td>3.3</td>
</tr>
<tr>
<td>ATRIAS [22]</td>
<td>1.13</td>
<td>62</td>
</tr>
<tr>
<td>2D-DURUS [7]</td>
<td>0.63</td>
<td>31.5</td>
</tr>
<tr>
<td>Cornell Biped [6]</td>
<td>0.2</td>
<td>13</td>
</tr>
<tr>
<td>Cornell Ranger [5]</td>
<td>0.19</td>
<td>9.9</td>
</tr>
<tr>
<td>DURUS</td>
<td>1.61</td>
<td>79.5</td>
</tr>
</tbody>
</table>
with the “control in the loop” leg design morphology that determined the configuration of passive compliant elements. The mechanical design was utilized in a theoretical design of stable walking gaits which exploit the full body dynamics of the robot. To stabilize the resulting trajectories on the robot when implemented experimentally, a preliminary heuristic feedback control strategy was presented. The resulting walking achieved on DURUS was shown to be stable and achieved consistently low electrical costs of transport while preserving a majority of the walking behaviors which give formal guarantees for the ideal system.

ACKNOWLEDGMENT

The authors would like to thank SRI for the design, development, and testing of DURUS, others in the AMBER Lab, Jessy Grizzle, and our collaborators in Dr. Jonathan Hurst’s Dynamic Robotics Laboratory at Oregon State University.

REFERENCES


