Trunk Stabilization of Multi-legged Robots Using On-line Learning via a NARX Neural Network Compensator

Brian R. Cairl[1], Farshad Khorrami[1]

***Abstract*— The objective of this work is to achieve disturbance rejection and constant orientation of the trunk of a multi-legged robot. This is significant when payloads (such as cameras, optical systems, armaments) are carried by the robot. In particular, this paper presents an application of an on-line learning method to actively correct the open-loop gait generated by a central pattern generator (CPG) or a limit-cycle method. The learning method employed is based on a Nonlinear Autoregressive Neural Network with Exogenous inputs (NARX-NN)– a recurrent neural network architecture typically utilized for modeling nonlinear difference systems. A supervised learning approach is used to train the NARX-NN. The input to the neural network includes states of the robot legs, trunk attitude and attitude rates, and foot contact forces. The neural network is used to estimate the total torque imparted on the robot. The learned effects of the internal forces and disturbances are then applied in an inverse dynamics/computed torque controller, which is utilized to achieve a stable trunk (i.e., a constant orientation of the trunk). The efficacy of the proposed approach is shown in detailed simulation studies of a quadruped robot.**

I. INTRODUCTION

Legged robotic systems have long employed motion controllers based on limit cycle oscillators and, more recently, Central Pattern Generators (CPGs) for the purpose of generating bio-mimetic gaits [1]–[9]. Since these motion control methods are open-loop motion planners they often require the use of auxiliary control mechanisms to ensure gait and system stability. Namely, methods which consider the zero-moment point and center of gravity of a legged system are often used when designing controllers to stabilize oscillator-driven gaits, as summarized in [10].

Developments in CPG-based gait controllers have led to the incorporation of "reflexive" feedback mechanisms aimed at correcting foot-placement during gaiting on uneven or slippery terrain. One such approach involves active compliance to each leg by directly modifying CPG oscillator parameters using aggregate joint feedback signals. In [3] and [11], CPG oscillators are modified by a using a single tuning parameter.

In this paper, we consider disturbance rejection of a multi-legged platform and achieving constant, level orientation of the trunk (*i.e.*, a stable platform), although other orientations could be considered. Disturbance rejection from the trunk sub-system of a legged platform has practical significance when carrying a payload (such as cameras, optical systems, armaments, etc.) rigidly fixed to the main body of the robot.

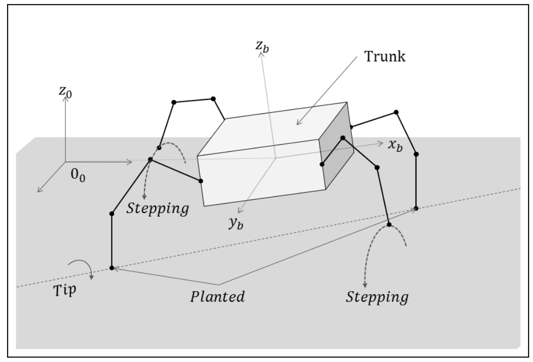


Fig. 1: Quadruped tipping about planted feet.

Disturbances are imparted upon the trunk during gaiting in two main ways: 1) instantaneous changes in force distribution when feet make and break contact with the ground, and 2) under-actuation that occurs during certain dynamic gaits. During dynamic trot gaits, for example, the state of contact between the feet and the ground is changed often so as to prevent the walking robot from tipping past a recoverable configuration. These gaits feature the utilization of two or fewer legs to support the trunk at any given time, causing the system to enter an under-actuated mode where the body is free to rock about the planted feet, as shown in Figure 1.

To achieve disturbance rejection on the trunk orientation and attain a fixed orientation, the proposed control methodology utilizes a Nonlinear Autoregressive Neural Network with Exogenous inputs (NARX-NN) as part of an active compensation mechanism. The network is used to estimate the system dynamics and, further, predict periodic disturbances in an on-line fashion. To present how the NARX-NN based controller is formulated, we will first examine the general-form dynamics of a quadruped system. Next, the NARX-NN compensator mechanism will be outlined along with an associated NARX-NN training regimen formulated with respect to the system dynamics. The compensator will then be formulated for use with a legged system implemented with a decentralized joint control architecture. Here, the compensator will be utilized to modify referential joint trajectories by way of a weighted sum between the original joint trajectories generated by a separate gaiting mechanism and a reference correction signal generated by the compensator.

The effectiveness of the proposed methodology will be shown through extensive simulation studies on a quadruped robot modeled after our in-house developed quadruped system (the BlueFoot Quadruped) during a CPG-driven trot gait. Results will highlight the robustness of the compensator

[1]All authors are with the Controls/Robotics Research Laboratory (CRRL) at the Polytechnic School of Engineering, 5 Metrotech Center, New York University, Brooklyn, NY 11201.

[2]Thank you to Dr. Prashanth Krishnamurthy for his input on this paper.

during gaits at various speeds. We also include results which depict the effect various of mixtures between the original CPG reference and NARX-NN compensator output signals. Lastly, some final remarks will be made about possible directions for enhancing the design of the compensator.

II. QUADRUPED DYNAMICS

We first consider a general, free-floating, four legged robotic system with m degrees of freedom per leg. This system is fully described by the state vector $\eta \in \mathbb{R}^{(6+4m)}$ and its dynamics are:

$$M(\eta)\ddot{\eta} + C(\eta, \dot{\eta})\dot{\eta} + G(\eta) + \Delta H = \tau + J^T(\eta) f_{ext} \quad (1)$$

where $M(\eta)$, $C(\eta, \dot{\eta})$, $G(\eta)$ and $J(\eta)$ represent the system mass matrix, Coriolis matrix, gravity matrix and Jacobian, respectively [12]. ΔH has been included as a lump term to account for dynamical uncertainties, such as friction or unmodeled coupling effects. Additionally, $f_{ext} = [f_{1,ext}^T, f_{2,ext}^T, f_{3,ext}^T, f_{4,ext}^T]^T \in \mathbb{R}^{24}$ represents a stacked vector of force-wrenches, $f_{i,ext} \in \mathbb{R}^6$, applied to the system through each i^{th} foot. The state vector, η, can be partitioned as follows: $\eta = [p_b^T, \theta_b^T, q^T]^T$ with $p_b \in \mathbb{R}^3$ and $\theta_b \in \mathbb{R}^3$ representing the position and orientation, respectively, of the quadruped's trunk in an arbitrarily placed world coordinate frame, and $q \in \mathbb{R}^{4m}$ is a vector of joint variables, m of which are contributed by each leg. $\tau \in \mathbb{R}^{(6+4m)}$ represents a vector of generalized torque inputs and takes the form $\tau = [0_{1x6}, \tau_q^T]^T$ where τ_q represents a set of torque inputs to each joint. It is important to note that the states we are most interested in controlling, p_b and θ_b, are not directly actuated, and must be controlled via composite joint motions.

A. Dynamics in State-Space Form and an Approximate Discrete-Time Realization

The dynamics in (1) can be realized in compact, state-space form by:

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= M^{-1}(z_1)(\tau + \Phi(z_1, z_2, f_{ext})) \\ \Phi(z_1, z_2, f_{ext}) &= J^T(z_1) f_{ext} - C(z_1, z_2) z_2 - G(z_1) - \Delta H \end{aligned} \quad (2)$$

where $z_1 = \eta$ and $z_2 = \dot{\eta}$. The notation $\Phi(z_1, z_2, f_{ext})$ is introduced for convenience and represents a composite dynamical term. This term will be referred to more compactly, as Φ, in the sections that follow.

The system dynamics are considered in an approximate, discrete-time form for use in NARX-NN network training, as follows:

$$\begin{aligned} z_{1,k+1} &= z_{1,k} + (e_{1,k}^{\Delta_s} + z_{2,k})\Delta_s \\ z_{2,k+1} &= z_{2,k} + M_{1,k}^{-1}(e_{2,k}^{\Delta_s} + \tau_k + \Phi_k)\Delta_s \end{aligned} \quad (3)$$

where $M_{1,k} = M(z_{1,k})$, $t = \Delta_s k$, and $\Delta_s \equiv (f_s)^{-1}$ with f_s defining a uniform sampling frequency in Hz. The terms $e_{1,k}^{\Delta_s}$ and $e_{2,k}^{\Delta_s}$ are used to explicitly account for system discretization errors, which vary with respect to the step-size, Δ_s. These discretization errors and system uncertainties will be compensated for by the NARX-NN learning mechanism.

B. Joint-Controller Dynamics

The motor dynamics driving each joint need to be considered since, in our case, the input to the servo motors at each joint is an angular reference command. In our compensation technique to follow, the combination of a neural network and inverse dynamics controller is used to produce an output torque which needs to be mixed with the joint trajectories provided by the CPG. In this paper, we will utilize a simple model of the motor dynamics at each joint to produce the required compensation torques by an equivalent joint trajectory to be followed by each motor. We consider the motors as simple torque generators of the following form:

$$\tau_q = k_s(q^r - q) \quad (4)$$

where $k_s > 0$ is a constant, scalar gain and q^r is a joint position reference. The servos we are utilizing to drive the leg joints of the BlueFoot quadruped have high-gain position feedback which allows us to model the motors as a static block which transforms reference trajectories to torque outputs. All of these servos are identical, and thus have identical gains. One could instead consider the full motor dynamics for computing reference positions given a desired torque. The simple model stated above was adequate for achieving the desired results.

III. NARX-NETWORK COMPENSATOR

A NARX-NN architecture is used in this controller because of its known effectiveness in approximating nonlinear difference systems and making multivariate time-series predictions [13]–[16]. Moreover, the NARX-NN is a natural fit for a problem of this nature where the dynamics being considered are both periodic and of a high enough complexity where a nonlinear approximation method is warranted. The parallel NARX-NN model, shown in Figure 2, is comprised of a feed-forward neural network whose input layer accepts a series of time-delayed system state values and network-output histories. The NARX-NN is trained to predict system states in the next time-instant from these inputs. Conveniently, NARX-NN training can be performed using standard BP because recurrence occurs between network inputs and outputs, and not within the hidden layers [17].

The NARX-NN is trained to capture the effects of forces and moments and dynamical couplings that act on the trunk so that an appropriate torque input to the joints is computed to reduce such effects on trunk orientation while performing the gate. This is achieved by considering the inverse dynamics corresponding to joint motion. Disturbances imparted upon the trunk during gaiting manifest in the term Φ, largely as a result of variations in f_{ext} and associated effects due to dynamical coupling. Because of this, the NARX-NN will learn an estimate for Φ, denoted $\hat{\Phi}$. The network is trained on-line using the standard incremental back-propagation (BP) algorithm with an adapted learning rate, γ [18], [19].

The success of this learning mechanism, as it applies to the presented controller, is predicated on the periodicity of the system dynamics during gaiting. Like any BP-trained neural network, repetition of similar input and output sets is

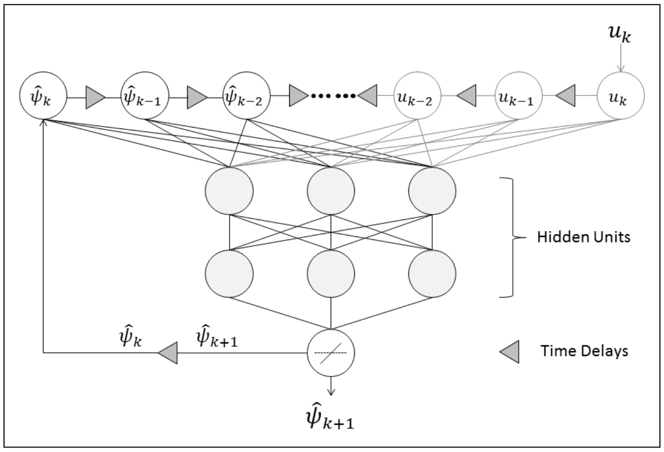


Fig. 2: Parallel NARX-NN model with a linear output layer.

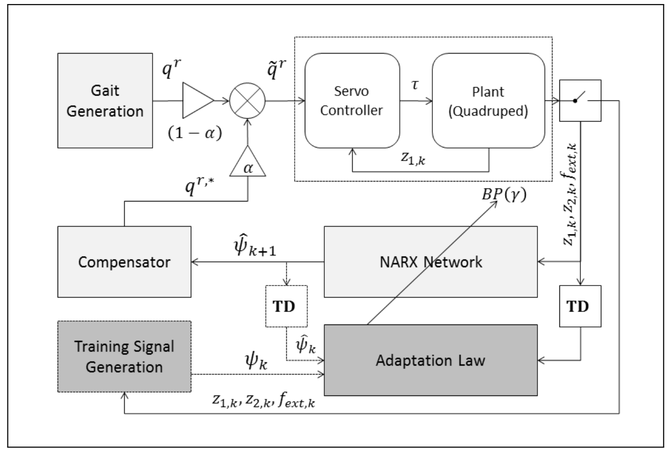


Fig. 3: Full system diagram with NARX-NN compensator.

paramount for successful network training and, by extension, prediction accuracy. It is assumed that this specification can be met given the inherently cyclic nature of the dynamics being estimated during gaited locomotion. Using the NARX-NN, the dynamical estimate $\hat{\Phi}_k$ is generated as a prediction of the sampled system dynamics, $\hat{\psi}_{k+1}$. The relationship between $\hat{\Phi}_k$ and the target prediction $\hat{\psi}_{k+1}$ will be made clear in the description of the network training signal given in (6). The general input-output relationship of the NARX-NN predictor, $\mathcal{N}$, is described as follows:

$$\begin{aligned} \hat{\psi}_{k+1} &= \mathcal{N}(\hat{\Psi}_k^N, U_k^N) \\ \hat{\Psi}_k^N &= [\hat{\psi}_k, \hat{\psi}_{k-1}, ..., \hat{\psi}_{k-N+1}] \\ U_k^N &= [u_k, u_{k-1}, ..., u_{k-N+1}] \end{aligned} \tag{5}$$

where U_k^N and $\hat{\Psi}_k^N$ are collections of N most recent samples of the network inputs, u_k, and the network output, $\hat{\psi}_k$, respectively. The NARX-NN input, u_k, represents a tuple $u_k = (z_{1,k}, z_{2,k}, f_{ext,k})$ whose components are the arguments of Φ at time instant k.

A. NARX-NN Training Regimen

The NARX-NN training signal is formulated to estimate Φ_k from the system dynamics. By (3), it can be seen that Φ_k can be estimated if $z_{2,k+1}$ can be predicted. We consider the following target prediction, ψ_{k+1}, defined by:

$$\psi_{k+1} = \tau_k - \hat{M}_{1,k}(z_{2,k+1} - z_{2,k})\Delta_s^{-1} = \Phi_k - e_{2,k}^{\Delta_s} \ . \tag{6}$$

This training signal formulation assumes that $\hat{M}_{1,k}$ represents $M_{1,k}$ exactly, which is likely not the case given the system's complexity. In the absence of a well-modeled $\hat{M}_{1,k}$, a constant symmetric $\hat{M}_{nom}$ will be picked such that $\hat{M}_{1,k} = \hat{M}_{nom} \forall k$. $\hat{M}_{nom}$ has the following structure:

$$\hat{M}_{nom} = \begin{bmatrix} \hat{M}_{bb} & \hat{M}_{bq} \\ \hat{M}_{qb} & \hat{M}_{qq} \end{bmatrix} \tag{7}$$

where $\hat{M}_{bb} \in \mathbb{R}^{6\times 6}$, $\hat{M}_{bq} = \hat{M}_{qb}^T \in \mathbb{R}^{6\times(4m)}$, and $\hat{M}_{qq} \in \mathbb{R}^{(4m)\times(4m)}$. It is particularly important that $\hat{M}_{bq} \neq 0$ to reflect some degree of coupling between the joint states q and the trunk states p_b and θ_b. In general, if $\hat{M}_{nom}$ should be selected to reflect the *average* system mass matrix over the range of configurations, z_1, seen during gaiting. This approximation has shown to be adequate from our results, and depends on the assumption that changes in $\hat{M}_{1,k}$ are small for the subset of values that z_1 takes on during a periodic gaiting sequence. Future improvements of this controller will involve the formulation of a separate estimator for $M(z_1)$, or a control/learning scheme with no direct dependence on $M(z_1)$.

Since $\hat{\psi}_{k+1}$ is non-causal, training is performed one time-step after a prediction is made using the input-output pair $\hat{\psi}_k$ and $\{\Psi_{k-1}^N,\ U_{k-1}^N\}$. Note that $\hat{\psi}_k$ can be calculated directly using (6) where all component signals are time-delayed by one time-step. Training can then be described by:

$$\psi_k \xrightarrow{BP(\gamma)} \mathcal{N}(\Psi_{k-1}^N, U_{k-1}^N) \tag{8}$$

where $\gamma_{min} < \gamma < \gamma_{max}$ is a learning rate adapted using a *bold-driver* update routine— a heuristic method for speeding up the convergence of BP-trained networks [20], [21]. This γ-update law is parameterized by $\beta \in (0,1)$ and $\zeta \in (0,1)$ which are selected to specify the amount by which γ increases or decreases per update, and γ_{min} and γ_{max} which are saturatation parameters. This scheme utilizes the current and previous mean-squared network output error values (MSE_k and MSE_{k-1}, respectively) to adjust γ as follows:

$$\gamma \leftarrow \begin{cases} \gamma(1-\beta) & \text{if } MSE_k > MSE_{k-1} \\ \gamma(1+\zeta\beta), & \text{otherwise.} \end{cases} \tag{9}$$

Since network training is being performed as an on-line routine, the effective mean-squared NARX-NN output error (MSE) is low-passed by a factor $\lambda \in (0,1)$. This update technique has been selected to ensure that outliers presented during training do not affect network learning updates as significantly as "nominal" training pairs. Each k^{th} MSE value is calculated one time-step after a prediction is made by:

$$MSE_k = \lambda \|\hat{\psi}_k - \psi_k\|_2^2 + MSE_{k-1}(1-\lambda). \tag{10}$$

B. Compensator Output

The control scheme is first presented with respect to the servo input torques, $\tau_{q,k}$, and formulated to achieve a level trunk characterized by $\theta_b = 0$, $\dot{\theta}_b = 0$. To formulate this controller, we first isolate the dynamical sub-system which corresponds to the un-actuated trunk orientation states by:

$$\ddot{\theta}_b = \Gamma_1 M^{-1}(z_1)(\Gamma_2 \tau_q + \Phi) \tag{11}$$

where

$$\Gamma_1 = [0_{3\times3}, I_{3\times3}, 0_{3\times(4m)}] \quad , \quad \Gamma_2 = [0_{(4m)\times6}, I_{(4m)\times(4m)}]^T$$

and $\Gamma_2 \tau_q$ is equivalent to the original system input, τ. In order to enforce a level platform with zero angular velocity, we seek a τ_q which emulates the proportional-derivative (P.D.) control law:

$$\ddot{\theta}_b = -K_p\theta_b - K_d\dot{\theta}_b \tag{12}$$

where K_p and K_d are constant gain matrices. Using this P.D. law and (11), we propose a least-squares solution for τ_q by:

$$\tau_q \approx -\left[\Gamma_1 M^{-1}(z_1)\Gamma_2\right]^{\dagger}\left[\Gamma_1 M^{-1}(z_1)\Phi + K_p\theta_b + K_d\dot{\theta}_b\right] \tag{13}$$

where $[*]^{\dagger}$ denotes the Penrose-Moore pseudo-inverse of $[*]$. Replacing all dynamical terms with their associated discrete-time equivalents, and Φ by the NARX-NN output $\hat{\Phi}_k = \hat{\psi}_{k+1}$, we apply (13) to arrive at the following required joint torque estimate:

$$\hat{\tau}_{q,k} = -\left[\Gamma_1 \hat{M}_{1,k}^{-1}\Gamma_2\right]^{\dagger}\left[\Gamma_1 \hat{M}_{1,k}^{-1}\hat{\psi}_{k+1} + K_p\theta_{b,k} + K_d\dot{\theta}_{b,k}\right] \tag{14}$$

where $\theta_{b,k}$ and $\dot{\theta}_{b,k}$ are samples of angular trunk position and rate, respectively.

The joint-reference compensator output, $\tilde{q}_k^r$, is formed as a weighted sum of the original gaiting trajectories, q_k^r, and a correction component, formed using (4) and (14), as follows:

$$\tilde{q}_k^r = (1-\alpha)q_k^r + \alpha\left(k_s^{-1}\left(\hat{\tau}_{q,k}\right) + q_{1,k}\right) \tag{15}$$

where $\alpha \in (0,1)$ is a uniform mixing parameter. The parameter α must be tuned with respect to the stability margins of the gait being compensated. The resultant $\tilde{q}_k^r$ is then applied to each joint controller in place of the original reference signal, q_k^r, generated by the gait controller. Selection of the parameter α is crucial for achieving good performance. The effects of α and its choice will be discussed in the ensuing section.

IV. EXPERIMENTAL PLATFORM AND SIMULATION STUDIES

A. Experimental Platform

We will be applying the NARX-NN compensator to our in-house developed robot, the BlueFoot Quadruped, shown in Figure 5. BlueFoot was designed for studies dealing with navigation on various types of terrain, as well as perception and 3D scene reconstruction. BlueFoot has 22-degrees of freedom with four revolute joints on each leg. BlueFoot's joints are controlled via smart-servo actuators which provide position, velocity and loading feedback. Each of BlueFoot's feet includes a binary-state contact sensor. Additionally, BlueFoot's trunk contains a 9-axis inertial measurement unit (IMU), GPS, and a vision sensor array consisting of a planar LIDAR and a stereo camera pair. We are also integrating load sensors into BlueFoot's feet for the proposed trunk stabilization algorithm implementation.

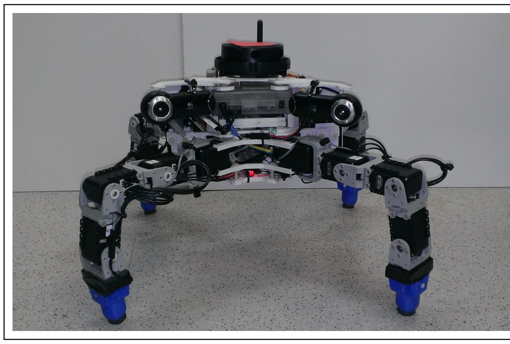


Fig. 4: The BlueFoot Quadruped.

B. Simulated Platform

The effectiveness of the proposed trunk stabilizer is validated through a series of detailed simulation studies. The simulator is implemented using the Open Dynamics Engine (ODE) [22] and models the BlueFoot platform with reasonable accuracy. Parameters of the simulated robot body, such as internal joint update gains, have been carefully tuned so that the simulator provides a high-fidelity representation of the physical platform.

In the ensuing simulation studies, to closely mimic the real platform and reduce simulated idealities, we have injected noise signals to various state measurements, e.g., angular position and velocities of the trunk. On the actual platform we are utilizing IMU measurements to estimate trunk pose via an extended Kalman filter.

Although contact forces on each foot are accessible, we estimate the force at each foot using a combination of trunk 3-axis accelerometers and foot contact data. Assuming a rigid system and a uniform distribution of forces to each planted foot, a rough estimate of the force applied to each i^{th} planted foot, $\hat{f}_i$, can be generated by:

$$\hat{f}_i = m_T\mu_i\left(\ddot{p}_b - \vec{g}\right) / \sum_{j=1}^{4}\mu_j \tag{16}$$

where m_T represents the total system mass; $\mu_i \in \{0,1\}$ is the contact state of the i^{th} foot (a value $\mu_i = 1$ represents contact); $\vec{g}$ is the gravity vector; and $\ddot{p}_b$ is the trunk acceleration in the world frame. Ideally, the measurement of f_i would be obtained via a 3-axis force-torque sensor placed at each foot.

C. Simulation Studies

In the simulations, the NARX-NN compensator is applied to the quadruped as it executes a stable CPG-driven trot gait

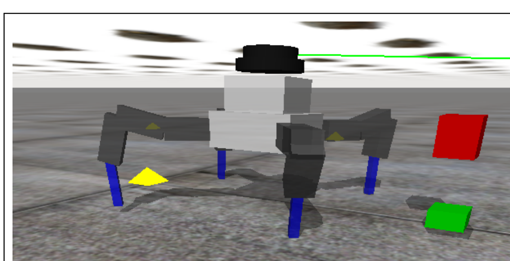


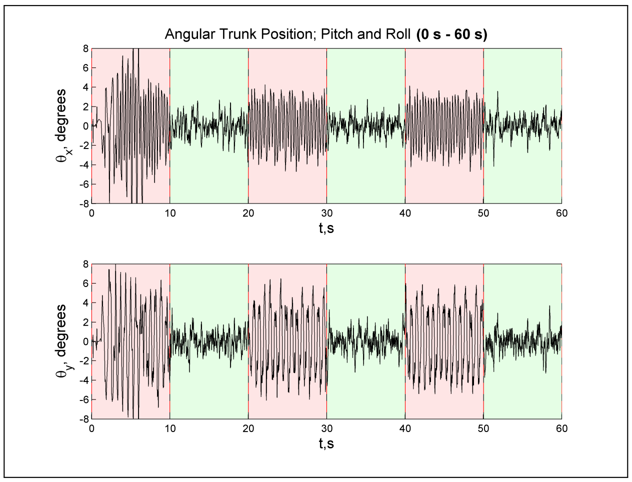
Fig. 5: The BlueFoot Quadruped simulator during gaiting.

depicted in Figure 1. In these trials, gaiting frequency is adjusted accordingly to achieve particular forward speeds.

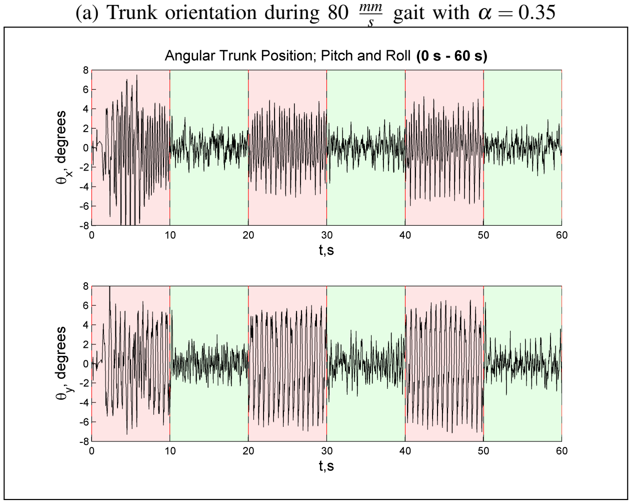
NARX-NN parameters are fixed for all trials with learning-rate parameters set to $\beta = 0.0001$, $\zeta = 0.0005$ and $\lambda = 0.01$. The NARX-NN is configured with two hidden layers containing 50 neurons each. Each input and hidden-layer neuron is modeled using a symmetric sigmoid activation function. Output layer neurons are modeled using linear activation functions to avoid output-scaling saturation issues. Figure 6 exemplifies the convergence of the NARX-NN.

All simulated trials are performed over a period of 60 seconds each. During the first 10 seconds of each simulation, the robot moves from sitting position to a standing position and initiates walking. During each simulation period, the NARX-NN compensator is activated (not training) and deactivated (training) every 10 seconds. Figure 7 depicts initial set of simulation results showing the effect of varying the mixing parameter $\alpha \in \{0.125, 0.25, 0.35\}$. For all such trials, the robot performs a trot-gait which achieves a forward speed of 60 $\frac{mm}{s}$. We expect that as α increases, the compensator will have greater authority over trunk stabilization. From these results, we observe that for all α, disturbance magnitude is decreased to some extent. However, for smaller α, the compensator is less effectual due to the fact that it has less authority over joint reference signals. From the results in Figure 7 **c)**, we see that the compensator improves pitch stability by more than roughly 50% and roll stability by more than 60%. Figure 6 shows the compensator's performance at higher gaiting speeds of 80 $\frac{mm}{s}$ and 100 $\frac{mm}{s}$. Here the controller improves both pitch and roll by nearly 50% and 40% of the uncompensated signal magnitude, respectively. Note that oscillations are still present even when the compensator is active. This is due to the fact that the NARX-NN learns to *generalize* the disturbances as a function of dynamical states/inputs, but is inherently unable to exactly predict disturbance effects. Here, we are interested in decreasing the magnitude of trunk oscillations, as the compensator is shown to achieve with obvious success.

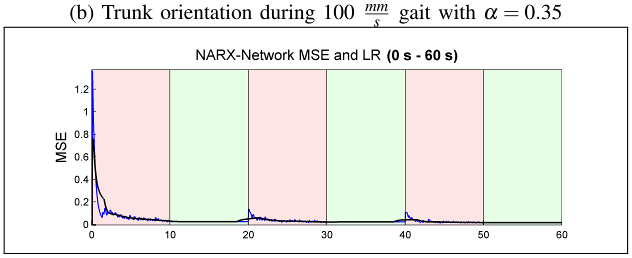
As may be inferred from the above observations, tuning the parameter α to achieve the desired performance is crucial. As the parameter α gets smaller (approaching zero), one recovers the original stabilized CPG generated gait (i.e., a CPG gait properly mixed with ZMP) and therefore loss of trunk stabilization. As α approaches one, the stabilized CPG generated gait is no longer utilized and the NARX network is generating the total gait for the robot. Since the NARX network is trained to attain a constant orientation of the trunk and reduces disturbances to the trunk, the generated gait does not take into account efficiency or stability of the gait. Therefore, it is crucial that a proper α (i.e., mixing of the CPG gait and the corrected gait) be chosen. To this extent, from our simulation studies, a value of 0.35 for α seems to be very effective. The gait corresponding to $\alpha > 0.375$ creates inefficient or unusable gaits for the robot (i.e., the forward movement is not achieved). In a sense, the parameter α reflects the stability margin of the properly stabilized CPG gait. One may consider a cost function to appropriately optimize the parameter α. Nevertheless, we believe our simulation studies depict stabilization results generated using a close-to-optimal choice of α.



(a) Trunk orientation during 80 $\frac{mm}{s}$ gait with $\alpha = 0.35$



(b) Trunk orientation during 100 $\frac{mm}{s}$ gait with $\alpha = 0.35$

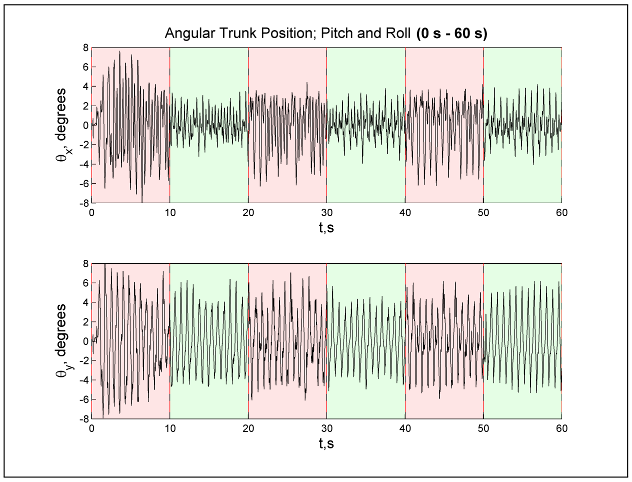


(c) NARX-NN prediction error during gait at 100 $\frac{mm}{s}$ with $\alpha = 0.35$

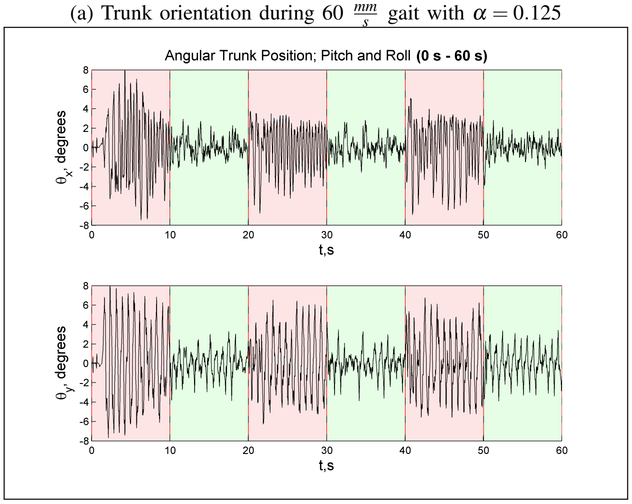
Fig. 6: Trunk orientation at various speeds. Dark-shaded regions depict when the compensator is not active.

V. CONCLUSION

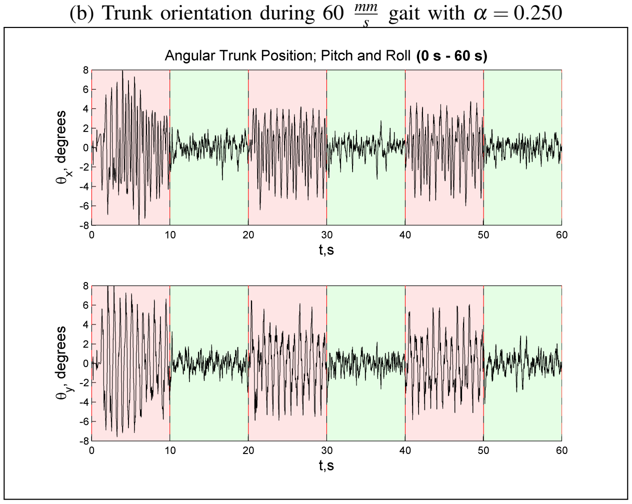
In this paper, a neuro-compensator mechanism based on a NARX-model Neural Network was presented to reject angular trunk state disturbances while executing periodic gait sequences. This control technique has been shown to be effective in suppressing platform disturbances, which suggests that it is extendable to control tasks which require steady articulation of a legged robot's main body. Further studies will aim to improve the degree of disturbance-rejection offered by the compensator and will test its effectiveness during a task in which the main body (trunk) is to be articulated over some desired trajectory. We are looking into further improvements of the neural network compensation scheme which reduces dependency on knowledge of the system mass matrix.



(a) Trunk orientation during 60 $\frac{mm}{s}$ gait with $\alpha = 0.125$



(b) Trunk orientation during 60 $\frac{mm}{s}$ gait with $\alpha = 0.250$



(c) Trunk orientation during 60 $\frac{mm}{s}$ gait with $\alpha = 0.350$

Fig. 7: Trunk orientation for varying α. Dark-shaded regions depict when the compensator is not active.

References

[1] Kiyotoshi Matsuoka. Sustained oscillations generated by mutually inhibiting neurons with adaptation. *Biological Cybernetics*, 52(6):367–376, 1985.

[2] J.J. Collins and Ian Stewart. Hexapodal gaits and coupled nonlinear oscillator models. *Biological Cybernetics*, 68(4):287–298, 1993.

[3] G. Endo, J. Morimoto, J. Nakanishi, and G. Cheng. An empirical exploration of a neural oscillator for biped locomotion control. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE Int. Conf. on*, volume 3, pages 3036–3042 Vol.3, April 2004.

[4] L. Righetti and A.J. Ijspeert. Programmable central pattern generators: an application to biped locomotion control. In *Proceedings 2006 IEEE International Conference on Robotics and Automation*, pages 1585–1590, May 2006.

[5] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural networks*, 21(4), 2008.

[6] Vitor Matos and Cristina P. Santos. Omnidirectional locomotion in a quadruped robot: A CPG-based approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3392–3397, Oct 2010.

[7] M. Ajallooeian, S. Pouya, A Sproewitz, and A.J. Ijspeert. Central Pattern Generators augmented with virtual model control for quadruped rough terrain locomotion. In *IEEE International Conference on Robotics and Automation*, pages 3321–3328, May 2013.

[8] Meng Yee (Michael) Chuah Park, Hae-Won and Sangbae Kim. Quadruped Bounding Control with Variable Duty Cycle via Vertical Impulse Scaling. 2014.

[9] Y. Fukuoka, H. Yasushi, and F. Takahiro. A simple rule for quadrupedal gait generation determined by leg loading feedback: a modeling study. *Sci. Rep.*, 5, 2015.

[10] Tedrake R. Kuindersma S. Wieber, P.-B. Modeling and Control of Legged Robots. In Siciliano and Khatib, editors, *Springer Handbook of Robotics, 2nd Ed*, Lecture Notes in Control and Information Sciences, chapter 48. Springer Berlin Heidelberg, 2015.

[11] Y. Fukuoka, H. Kimura, Y. Hada, and K. Takase. Adaptive dynamic walking of a quadruped robot 'Tekken' on irregular terrain using a neural system model. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 2037–2042 vol.2, Sept 2003.

[12] P.-B. Wieber. Holonomy and Nonholonomy in the Dynamics of Articulated Motion. In Moritz Diehl and Katja Mombaur, editors, *Fast Motions in Biomechanics and Robotics*, volume 340 of *Lecture Notes in Control and Information Sciences*, pages 411–425. Springer Berlin Heidelberg, 2006.

[13] Tsungnan Lin, B.G. Horne, P. Tino, and C.L. Giles. Learning long-term dependencies in NARX recurrent neural networks. *Neural Networks, IEEE Transactions on*, 7(6):1329–1338, Nov 1996.

[14] Grant P. M. Chen S., Billings S. A. Non-linear system identification using neural networks. In *Int. Journal of Control*, volume 51 of *Lecture Notes in Control and Information Sciences*, pages 1191–1214. Taylor and Francis, 1990.

[15] Salah El Hihi and Yoshua Bengio. Hierarchical Recurrent Neural Networks for Long-Term Dependencies, 1996.

[16] S. A Billings. *Nonlinear system identification : NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley and Sons Ltd, 2013.

[17] Oliver Nelles. Neural Networks with Internal Dynamics. In *Nonlinear System Identification*, pages 645–651. Springer Berlin Heidelberg, 2001.

[18] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of Research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[19] David E. Rumelhart, Richard Durbin, Richard Golden, and Yves Chauvin. Backpropagation. chapter Backpropagation: The Basic Theory, pages 1–34. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1995.

[20] Roberto Battiti. First and Second-Order Methods for Learning: between Steepest Descent and Newton's Method. *Neural Computation*, 4:141–166, 1992.

[21] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis. Improving the Convergence of the Backpropagation Algorithm Using Learning Rate Adaptation Methods. *Neural Computation*, 11(7):1769–1796, oct 1999.

[22] Russel Smith. *Open Dynamics Engine, http://www.ode.org/*.