Trunk Stabilization of Quadruped Robots Using an On-line NARX-Network Compensator

Brian Cairl[1]

***Abstract*— This paper addresses a method of adaptive body orientation control for quadruped robots mobilized using open-loop gait control methods, such as Central Pattern Generators (CPG). The controller at hand is based on a feed-foward, inverse dynamics routine. Controller adaption is achieved using a Nonlinear Autoregressive eXogenous (NARX)-model Neural Network – a recurrent neural network architecture typically utilized for modeling nonlinear difference systems. Here, the NARX Network is utilized to make predictions about the system dynamics, which are assumed to be periodic during the execution of a cyclic gait. Furthermore, the controller serves as an on-line method for learning and canceling disturbances imparted upon angular trunk states. This control method is particularly applicable to legged systems which articulate trunk-mounted vision sensors, such as cameras and laser range-finders, by reconfiguring thee kinematic state of the trunk while executing a gait sequence. By canceling trunk distrubances, the fidelity of vision sensors meausurements can be enhanced by removing noise before it propogates to the aforementioned sensor units.**

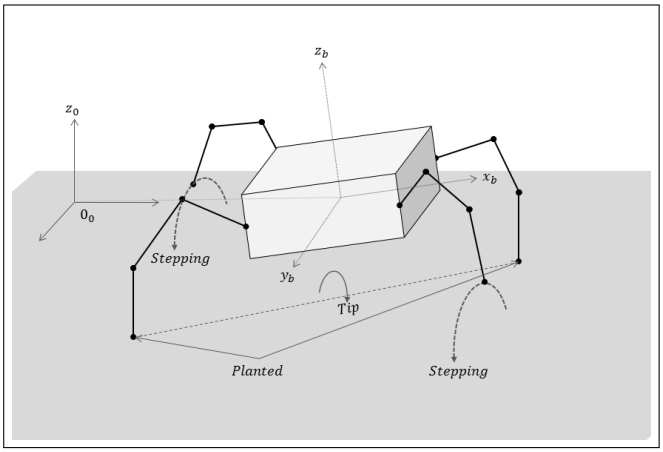


Fig. 1: Quadruped Tipping About Planted Feet During A Trot Gait

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, in essence, open-loop motion planners they do not guarantee stability and cannot be applied directly without further consideration of the legged system's dynamics. Moreover, proper implementation of the aforementioned methods of locomotion requires the use of auxiliary control mechanisms which can adapt the system to disturbances and changes in environmental conditions.

Namely, developments in CPG-based gait controller have led to the incorporation "reflexive" feedback mechanisms aimed at correcting foot-placement and trunk posture during gaiting [3], [10]. These methods are meant to further extend to similarity of legged systems to their biological counterparts, and hinge on fixed-point control methods covered in detail in [11]. However, few gait controllers of this nature, however, utilize an on-line learning mechanism to handle the task of achieving system stability, namely angular trunk stability.

Disturbance rejection from the angular body states of is of practical importance for larger legged systems with many sensors rigidly fixed to their trunk sections. By extension, the trunk of a legged system could be used to pitch and roll sensor units, like camera's and laser distance sensors, as will be done with our *BlueFoot* quadruped system, shown in [NEED FIG] . Performing such a task during gaiting certainly demands proper control stability control of angular trunk states. Without compensation vision sensor measurements will likely be disturbed by vibrations caused by instantaneous changes in force distribution, namely, when feet make and break contact with the ground. Additionally, certain dynamic gaits during which two or fewer legs are in contact with the ground at any given time cause the system to enter under-actuated state where the trunk is free to rock about the planted feet, as shown in Figure 1. These gaits however, are advantageous because they typically allow quadruped systems to attain higher land speeds than with static creep-gaits [*] . To address the problem of trunk-state disturbance rejection during CPG-generated, non-static gaits, this paper will present a trunk-stabilization routine that utilizes a Nonlinear Autoregressive eXogenous (NARX)-model Nueral Network to learn and predict the quadruped system during gaiting and predict periodic disturbances.

The use of a NARX-model Neural Network is particularly important feature of the control method being presented, as it has been shown that network models have an affinity for approximating nonlinear difference systems and make chaotic time-series predictions [12]–[15]. This provides a natural fit to a problem such as this, where the dynamics at are both periodic and of a high enough complexity where an approximation method is certainly warranted.

This paper will first examine the general-form dynamics for a legged system, and a first-order discrete-time counterpart. Next, the nuero-compensator mechanism will be outlined, and the NARX-Network training regimen will be

described with respect to the system dynamics. This will also include a section about how the compensator output is applied to correct joint reference signals for when the compensator is applied to a legged system implemented with a decentralized joint control architecture. Lastly, results showing the effectiveness of the compensator, as it is applied to a quadruped platform (the *BlueFoot* Quadruped) executing a CPG-drive trot gait, will be presented. Results will highlight the robustness of this approach, even when applied to gaiting over a surface with unperceived

II. QUADRUPED DYNAMICS

We begin by considering a general, free-floating, four legged robotic system with m degrees of freedom per leg. This system is fully described by the state vector $\eta \in \mathbb{R}^{(6+4m)\times 1}$ and evolves as per the following dynamics:

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

where $M(\eta)$, $C(\eta,\dot{\eta})$, $G(\eta)$ and $J(\eta)$ represent the system mass matrix, Coriolis matrix, gravity matrix and Jacobian, respectively [16]. ΔH has been included as a lump term for any dynamical uncertainties, such as friction or unmodeled coupling effects. Additionally, $f_{ext} \in \mathbb{R}^{6\times 1}$ represents the total external force-wrench applied to the system.

The state vector, η, can be divided into the following sub components: $\eta = [p_b^T, \theta_b^T, q^T]^T$. $p_b \in \mathbb{R}^{3\times 1}$ and $\theta_b \in \mathbb{R}^{3\times 1}$ which represent the position and orientation, respectively, of the quadruped's trunk, in an arbitrarily placed world coordinate frame [11]. $q \in \mathbb{R}^{4m\times 1}$ is a vector of joint variables, m of which are contributed by each leg.

$\tau \in \mathbb{R}^{(6+4m)\times 1}$ represents a vector if generalized torque inputs and takes the form $\tau = [0_{1x6}, \tau_q]^T$, where τ_q represents the torque inputs to the joints. 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 through a composite of leg 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:

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

where $z_1 = \eta$ and $z_2 = \dot{\eta}$. The notation $\Phi(z_1, z_2, f_{ext})$ is invented for convenience when dealing with $M(z_1)$, $C(z_1, z_2)$, $G(z_1)$, $J(z_1)$ and ΔH as a composite dynamical term. This term will be referred to, simply, as Φ in the sections that follow.

Because the routine present in this paper is realized as a digital controller which, it is convenient to deal with the system dynamics, (2), as a discrete-time approximation of the following form:

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

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

B. Joint-Controller Dynamics

An important part of the control algorithm at hand is its extensibility to systems that have been implemented with decentralized joint controllers. In the case where a decentralized joint control architecture is used, control over the torques applied at each is typically not explicit. To deal with this an inverse joint control model (or some adequate approximation) is needed for proper interfacing with the model-based controller.

In many cases, the controller dynamics of a revolute joint-position controllers can be described as a simple proportional (P), or P.D. control schemes. The BlueFoot platform, for example, utilizes 16 smart-servo controllers that can be described via a P-control scheme. Using the introduced notations, BlueFoot's joint control scheme takes the follow aggregate form:

$$\tau = K_s(z_1^r - z_1) = \begin{bmatrix} 0_{6\times 6} & 0 \\ 0 & k_s I_{(4m)\times(4m)} \end{bmatrix}(z_1^r - z_1) \qquad (4)$$

where $k_s > 0$ is a constant, scalar gain. For the remainder of this paper, we will assume a decentralized joint-position control regime of this form. The control method at hand, however, is certainly not limited to platforms with this specific joint control architecture. With slight modifications, it is conceivable that this control technique could be extended to any other decentralized joint control architecture with invertible or least-squares invertible controller dynamics.

III. NARX-NETWORK COMPENSATOR

A feed-forward, inverse dynamics routine forms the basis for the control scheme at hand. Using the previously introduced notations, the inverse dynamics controller takes on the following general form:

$$\tau = \hat{M}(z_1)[\dot{z}_2^r + K_1(z_1^r - z_1) + K_2(z_2^r - z_2)] + \hat{\Phi} \qquad (5)$$

where $\hat{M}(z_1)$ and $\hat{\Phi}$ are approximations of M and Φ, respectively; and K_1 and K_2 are constant, square gain matrices. Since the disturbances generated during gaiting are present in the term Φ as a result of f_{ext}, and associated propagation effects due to dynamical coupling, the control algorithm at hand will focus on learning its associated estimate, $\hat{\Phi}$.

In the controller at hand, the role of an estimator for $\hat{\Phi}$ is assumed by NARX-model neural network, trained online using the standard incremental back propagation (BP) algorithm with an adapted learning rate, γ [17], [18].

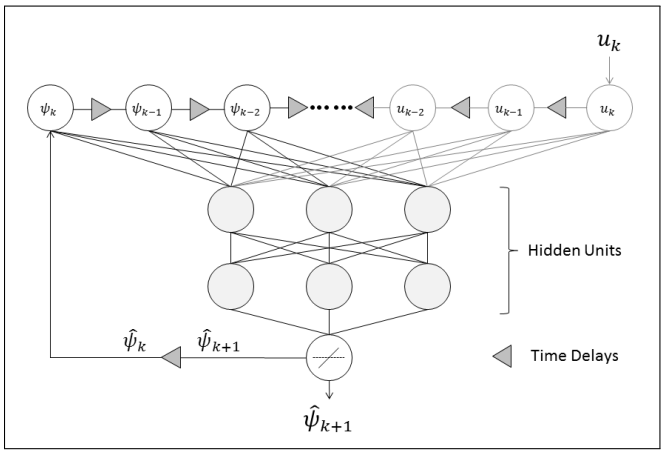


Fig. 2: Parallel NARX-Network Model with a Linear Output layer

As previously mentioned, a network of this particular configuration has been chosen for its known effectiveness in modeling nonlinear difference equations and application to chaotic time-series prediction. Given that the NARX-network employs histories of presented input and output signals to make predictions, the success of this learning mechanism is predicated on the of periodicity of the dynamics at hand, as the repetition of similar input and output sets is paramount for successful network training and, by extension, prediction stability. It is assumed that this specification can be met given the inherently cyclic nature of gaited locomotion. $\hat{\Phi}$ is generated by way of forward *look-ahead* of the sampled system dynamics, ψ_{k+1}. The relationship between $\hat{\Phi}_k$, the estimate $\hat{\Phi}$ at time instance k, and the network prediction $\hat{\psi}_{k+1}$ will be made clear in the description of the network training signal.

The general input-output relationship of the NARX network predictor, $\mathcal{N}$, is described as follows:

$$\begin{aligned} \hat{\psi}_{k+1} &= \mathcal{N}(\hat{\Psi}_k^N, U_k^N) \\ \Psi_k^N &= [\psi_k, \psi_{k-1}, ..., \psi_{k-N+1}] \\ U_k^N &= [u_k, u_{k-1}, ..., u_{k-N+1}] \end{aligned} \tag{6}$$

where $\hat{\Psi}_k^N$ and U_k^N are collections of N most recent samples of the network inputs and bouts, $\hat{\psi}$ and u, respectively. In this case each network input entry, u_k, represents a tuple $u_k = (z_{1,k}, z_{2,k}, F_{ext,k})$ whose components are related to the arguments of the true Φ at time instance k. The mos prominent difference between these input sets that $F_{ext,k} \in \mathbb{R}^{12\times 1}$ is not equivalent to the force-wrench f_{ext}. Instead, $F_{ext,k}$ is stacked vector of translational forces, $f_i \in \mathbb{R}^{3\times 1}$, applied to each i^{th} end-effector, *i.e.*, $F_{ext} = [f_1^T, f_2^T, f_3^T, f_4^T]^T$. Here, we are more interested in learning a mapping between the variable distribution of applied forces, which is subject to instantaneous, periodic changes. Additionally, sensing the translational force applied to each foot is much more straightforward in an, implementation sense, then sensing the full force-wrench, which also contains a torsional component.

A. Training the NARX Network

A formulation for the NARX network training signal for the begins by isolating the unknown portions of the system dynamics, Φ_k, which can be estimated using predictions which incorporate the system's state at time instance $k+1$, namely $z_{2,k+1}$. This look ahead is made possible by the predictive capabilities of the NARX network.

$$\Psi_{k+1} = \tau_k - \hat{M}_{1,k}(z_{2,k+1} - z_{2,k})\Delta_s^{-1} = \Phi_k - e_{2,k}^{\Delta_s} \tag{7}$$

It should be noted that in addition to learning to predict the system dynamics, the training signal at hand teaches the network to absolve the discretization error, $e_{2,k}^{\Delta_s}$, when applied to a controller based on the discrete time model described by (3).

This formulation (7) assumes that $\hat{M}_{1,k}$ represents $M_{1,k}$ to some exactness, which is likely not the case given the system's complexity. In the absence of $\hat{M}_{1,k}$ in a more accurate form, 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:

$$\hat{M}_{nom} = \begin{bmatrix} \hat{M}_{bb} & \hat{M}_{bq} \\ \hat{M}_{qb} & \hat{M}_{qq} \end{bmatrix} \tag{8}$$

where $\hat{M}_{nom} \in \mathbb{R}^{(4m+6)\times(4m+6)}$ is a symmetric, constant matrix with component blocks $\hat{M}_{bb} \in \mathbb{R}^{6\times 6}$, $\hat{M}_{bq} = \hat{M}_{qb}^T \in \mathbb{R}^{6\times(4m)}$, and $\hat{M}_{qq} \in \mathbb{R}^{(4m)\times(4m)}$. It is particularly important that $M_{bq} \neq 0$ to reflect some degree of coupling between the joint states q and the trunk states p_b and θ_b. In general, $\hat{M}_{nom}$ should be selected to reflect the *average* system mass matrix over the range of configurations, z_1, seen during gaiting.

To train the predictor, the network must wait until time instance $(k+1)$ when a "true" value for $\hat{\psi}_{k+1}$, becomes known. Since ψ_k is a completely causal signal, it can be calculated directly. The network is, then, trained using the set of previous input, $\{\ \Psi_{k-1}^N,\ U_{k-1}^N\}$, and the true output training signal, ψ_k, as follows:

$$\psi_k \overset{BP_\gamma}{\rightarrow} \mathcal{N}(\Psi_{k-1}^N, U_{k-1}^N) \tag{9}$$

where γ is a learning rate adapted using a *bold-driver* update routine, parameterized by $\beta \in (0,1)$ and $\zeta \in (0,1)$, which represent an exponential learning rate modification and penalization-bias factors, respectively. This scheme is heuristic methods for speeding up the rate of convergence of back-propagation training regimes [19], [20]. The bold driver routine updates γ according to the mean-squared output error values $(MSE)_k$ and $(MSE)_{k-1}$ as follows:

$$\gamma \leftarrow \begin{cases} \gamma(1-\beta) & \text{if } (MSE)_k > (MSE)_{k-1} \\ \gamma(1+\zeta\beta), & \text{otherwise} \end{cases} \tag{10}$$

In cases where network training is being performed on-line as an incremental routine, the effective mean-squared error of the network output low-passed by a factor $\lambda \in (0,1)$. This is performed so as to ensure that outliers presented during training do not effect network learning rate updates as significantly. Moreover, network output error, $e_{\mathcal{N},k}$ and its

associated MSE values are calculated one time instance after each prediction is made by:

$$\begin{aligned} e_{\mathcal{N},k} &= \hat{\psi}_k - \psi_k \\ (MSE)_k &= \lambda \|e_{\mathcal{N},k}\|_2^2 + (MSE)_{k-1}(\lambda - 1) \end{aligned} \tag{11}$$

B. Compensator Outputs

The controller scheme is first presented with respect to the general input torques, τ_k and is based on a discrete-time representation of an inverse dynamics controller. Instead of using an approximate dynamical model for the system, unknown system dynamics are substituted with the $\hat{\Psi}_{k+1}$ generated as the output of the network, $\mathcal{N}_k$. respectively.

$$\begin{aligned} \tau_k &= M_{1,k}[\Delta e_{2,k+1} + K_p e_{1,k} + K_p e_{2,k}] + \hat{\Psi}_{k+1} \\ e_{1,k} &= z^r_{1,k} - z_{1,k} \\ e_{2,k} &= z^r_{2,k} - z_{2,k} \\ \Delta e_{2,k+1} &= (z^r_{2,k+1} - z_{2,k})\Delta_s^{-1} \end{aligned} \tag{12}$$

where $z^r_{2,k+1}$ defines the velocity commands issued to each states during pre-planned gaiting sequence.

The control scheme input, τ_k, must now be tailored to the available system control inputs, τ_q. Given the actuator model (4), we will assumed that torques are controlled indirectly via supplied actuator reference positions, and will instead consider an outer loop controller for generating a set of "corrected" actuator positions $z^{r,*}_k$.

$$z^{r,*}_{1,k} = [K_s]^{\dagger}\left[M_{1,k}[\Delta e_{2,k+1} + K_p e_{1,k} + K_d e_{2,k}] + \hat{\Psi}_{k+1}\right] + z_{1,k} \tag{13}$$

where the $[K_s]^{\dagger}$ is the Penrose-Moore pseudo-inverse of the K_s. Given the structure of K_s, the compensator output $z^{r,*}_{1,k}$ is reduced to $z^{r,*}_{1,k} = [0_{1\times 6}, (q^{r,*}_{1,k})^T]^T$. Here is it assumed that the states $\dot{p}_b$ and $\dot{\theta}_b$ are bounded and the sub-system which governs the trunk-state dynamics is asymptotically stable provided that the all joint states are bounded during gaiting. Additionally, it is assumed all external forces which disturb the trunk states enter the system through the legs, *i.e.*, there no external forces being applied to the body states. Thus,

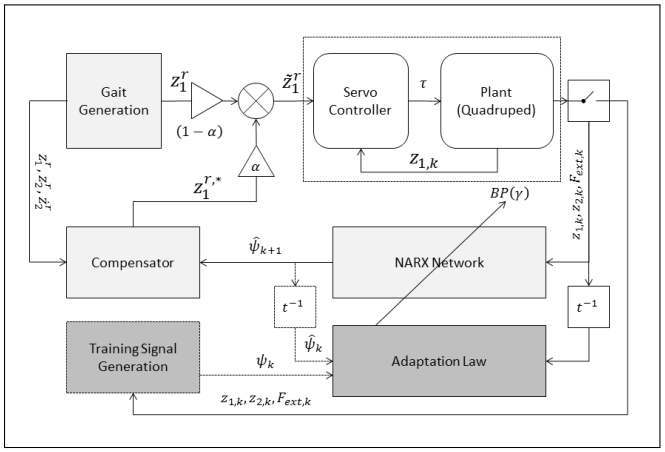


Fig. 3: Full system diagram with NARX-Compensator Mechanism

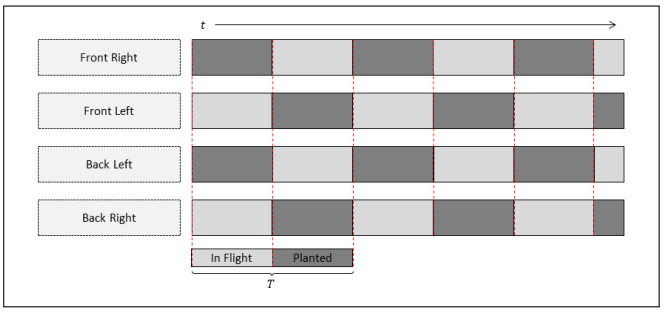


Fig. 4: Trot gait sequence

it is sensible to perform the necessary disturbance rejection control in the joint-space.

The correction signal, $q^{r,*}_k$ is combined with the original trajectory signal q^r_k as follows:

$$\tilde{q}^r_k \leftarrow (1-\alpha)q^r_k + \alpha(q^{r,*}_k) \tag{14}$$

where $\alpha \in (0,1)$ is a mixing parameter for combining the *a priori* reference trajectory vector q^r_k and the correction signal $q^{r,*}_k$.

IV. EXPERIMENTAL RESULTS

A. Experimental Target and Considerations

The Nuero-control algorithm presented is tested using a high-fidelity quadruped simulator, based on the Open Dynamics Engine (ODE) [21], has been designed for testing control algorithms used on the BlueFoot quadruped platform. The simulated BlueFoot robot is outfitted with 16-revolute joints (4 per leg) which are controlled with independent proportional feedback loops.Controller gains have been chosen to match the response of servo actuators used on the physical system. Additionally, angular position and velocity feedback are available for each joint. The platform is also outfitted with a 12-axis inertial measurement unit (IMU) and binary-state contact sensors on each of its feet. To be consistent with BlueFoot's sensor outfit, the simulator does not emulate full state feedback. The compensator is setup to perform as a reduced-state version of the originally present control scheme. In this reduced state representation, the trunk states p_b and $\dot{p}_b$ are ignored. This does not change the general form of the controller at hand. Ideally, $\dot{p}_b$ would be estimated using a combination of vision-based odometry measurements and, p_b would be estimated via a localization routine.

Secondly, the angular position state of the robot's trunk is estimated via an Extended Kalman Filter emulation. This emulation mixes the true trunk orientation signal, θ_b, with a low-passed, Gaussian white noise signal to create the effect of base-band drift.

Lastly, the force applied to each i^{th} foot is estimated using a combination accelerometer and foot-contact data. Assuming a rigid system and uniform distribution, a rough estimate of the force applied to each i^{th} planted foot, $\hat{f}_i$, can be generated by:

$$\hat{f}_i = \frac{\mu_i}{m_T \sum_{j=1}^{4} \mu_j}(\ddot{p}_b - \vec{g}) \tag{15}$$

where m_T is a scalar representing 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, these applied force vectors would be estimated using force-magnitude sensors placed on each foot.

The following results will shown that the aforementioned approximations do not compromise the performance of the compensator mechanism.

The compensator is applied to the simulated BlueFoot robot as it executes a CPG-driven trot gait depicted in 4. This gaiting pattern is produced using a set of CPG parameters borrowed from [*] . During this gait, there are two planted feet, and two feet in flight at any time. The gait period, T, is adjusted to achieve some desired land-speed. This gait is the default used for mobilizing the BlueFoot platform.

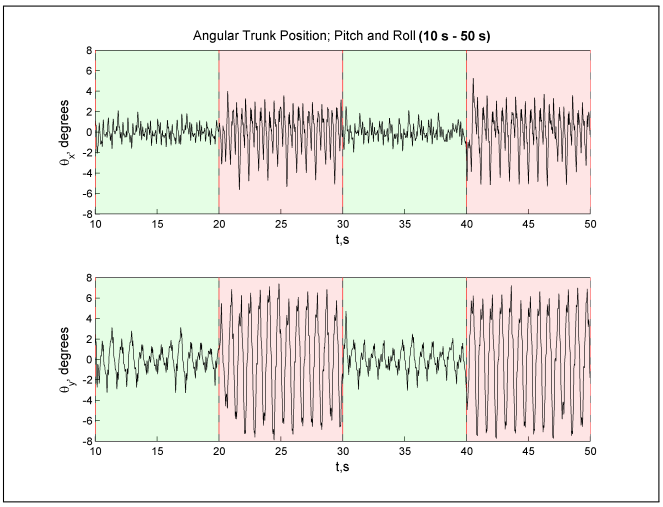
The remaining experimental parameters are set as follows: mixing parameter is set to $\alpha = 0.25$; learning-rate parameters are set to $\beta = 0.001$ and $\zeta = 0.005$. The NARX-Network is setup with two hidden layers containing 50 neurons each. Each neuron is modeled using a symmetric, $tanh(*)$, activation function. Output layer neurons are modeled using linear activation functions to avoid output-scaling saturation issues.

B. Simulated System Results

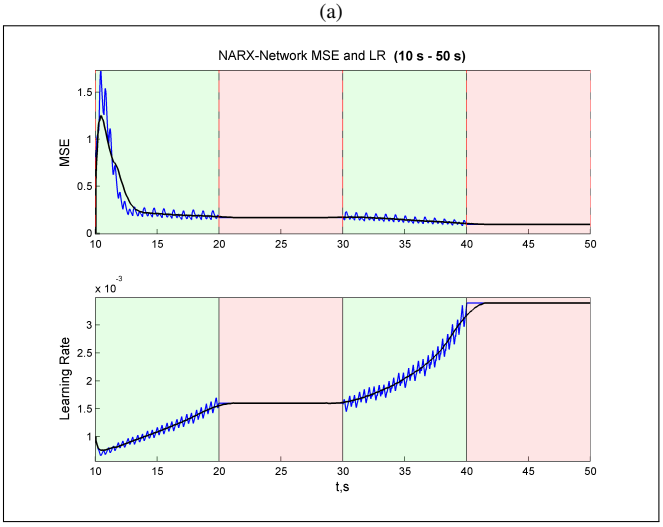
Figures 5 and 6 depict the roll and pitch trunk states during gaiting with alternating Neuro-compensator activity. Is is shown in Figure 5 that during a moderately-paced gait used to achieve an average land-speed of $65\frac{mm}{s}$, the compensator (active between $t \in (10,20)$ and $t \in (30,40)$) decreases roll and pitch deviation from the zero-state by more than 50%.

V. CONCLUSION

This paper has presented a neuro-compensator mechanism based on a NARX-model Network. The compensator is utilized to reject angular trunk state disturbance while executing periodic gait sequences.



(a)



(b)

Fig. 5: **a)** Trunk position **b)** Network MSE and Learning rate during gait for average ground speed $65\frac{mm}{s}$

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 International Conference 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. 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] Ajallooeian, M., Gay, S., Tuleu, A., Sprowitz, A, and Ijspeert, A.J. Modular control of limit cycle locomotion over unperceived rough terrain. pages 3390–3397, Nov 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] 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 *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 2037–2042 vol.2, Sept 2003.

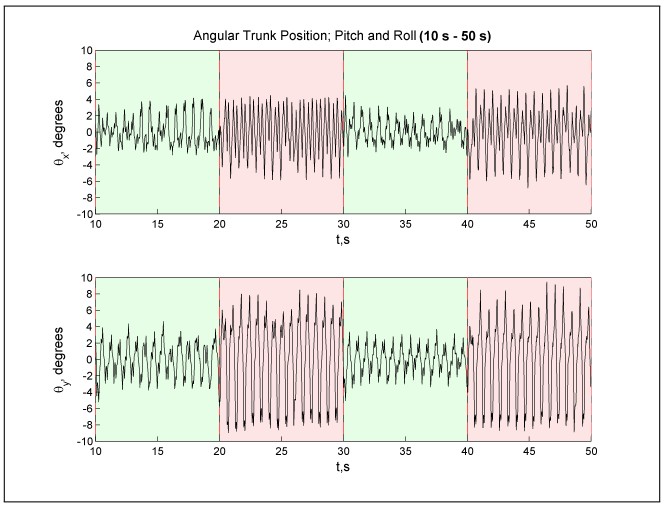
[11] Tedrake R. Kuindersma S. Wieber, P.-B. Modeling and control of legged robots. In Siciliano and Khatib, editors, *Springer Handbook of Robotics, 2nd Ed*, Lecture Notes in Control and Information Sciences, chapter 48. Springer Berlin Heidelberg, 2015.

[12] 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.

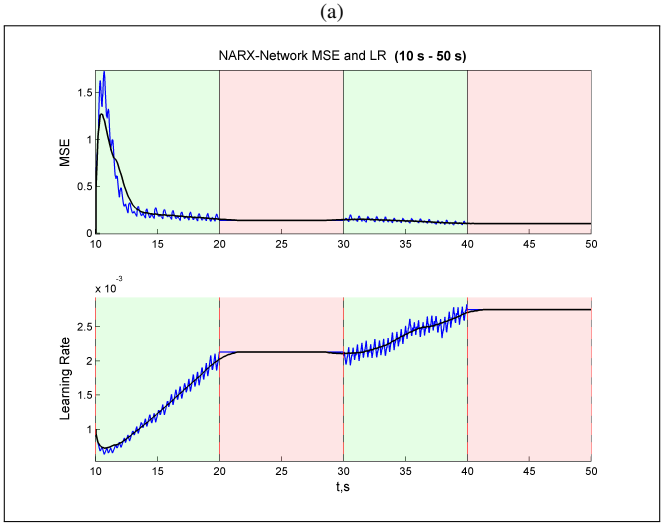
[13] Grant P. M. Chen S., Billings S. A. Non-linear system identification using neural networks. In *International Journal of Control*, volume 51 of *Lecture Notes in Control and Information Sciences*, pages 1191–1214. Taylor and Francis, 1990.

[14] Salah El Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies, 1996.

[15] S. A Billings. *Nonlinear system identification : NARMAX methods*



(a)



(b)

Fig. 6: **a)** Trunk position **b)** Network MSE and Learning rate during gait for average ground speed $80\frac{mm}{s}$

in the time, frequency, and spatio-temporal domains. John Wiley and Sons Ltd, 2013.

[16] 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.

[17] 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.

[18] 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.

[19] Roberto Battiti. First and second-order methods for learning: between steepest descent and newton's method. *Neural Computation*, 4:141–166, 1992.

[20] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis. Improving the convergence of the backpropagation algorithm using learning rate adaptation methods. *Neural Comput.*, 11(7):1769–1796, oct 1999.

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