Optimal Decentralized Gait Transitions for Snake Robots

Greg Droge, Magnus Egerstedt

***Abstract*—Snake robots are controlled by implementing gaits inspired from their biological counterparts. However, transitioning between these gaits often produces undesired oscillations which cause net movements that are difficult to predict. In this paper we present a framework for implementing gaits which will allow for smooth transitions. We also present a method to determine the optimal time for each module of the snake to switch between gaits in a decentralized fashion. This will allow for each module to participate in minimizing a cost by communicating with a set of modules in a local neighborhood. Both of these developments will help to maintain desired properties of the gaits during transition.**

I. Introduction

Snake robots have the ability to traverse through tunnels, over flat ground, swim, and even crawl up crevices (eg [1], [2]). However, this versatility comes at a cost. Snake robots are highly underactuated nonholonomic systems and the physics of snake locomotion are difficult to model as the terrain can be quite complex [2]. To reduce the complexity of controlling a robot snake, researchers have turned to nature for inspiration (eg [1], [2], [3]). Snakes use their entire body to execute gaits such as slither or sidewind which can often be modelled by a sinusoid wave which propagates down the length of the snake [4]. By abstracting the movement of a robot to execute these gaits, the dimensionality of the control problem is greatly reduced. These net motions can then, theoretically, be used with an array of motion planning algorithms (see for example [5]).

Yet, a major difficulty lies in the transition between gaits. Any type of unexpected oscillatory motion, which often occurs during such transitions, can produce undesired net movements. This is due to the complicated physical interaction with the snake and ground during movement. Therefore, in this paper we present a framework to reduce such unexpected oscillations.

There are two major sources of oscillations that we will seek to eliminate in this paper. The first comes from the implementation of these gaits using inherently time dependent sinusoid generators which do not provide a natural way to smoothly transition between gaits. Another major source of these undesired oscillations is the failure for the snake to maintain certain properties during transition. To address these two issues, we present a decentralized framework building on limit cycle methods to produce cyclic motion (eg [6]) as well as the theory of switch time optimization (eg [7]) which will allow for the gait transition to propagate down the snake while satisfying desired properties by minimizing a cost.

The remainder of the paper will proceed as follows. In the next section we will give a brief overview of how gaits for the snake robot have been modelled. In section III we will outline an approach for accomplishing these gaits using a decentralized control scheme. We will then present, in section IV, a decentralized switch-time optimal control scheme to determine when to switch gaits. We will end the paper by showing an example of this framework in section V and some concluding remarks in section VI.

Email: {gregdroge,magnus}@ece.gatech.edu.
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA.

II. Snake Gaits

As modeling snake locomotion can be difficult due to the fact that snake movement is complicated and the physics of the interaction forces in unknown environments can be virtually impossible to model, researchers have turned to nature to find a suitable abstraction (eg [1], [2], [3]). The authors in [4] showed that many gaits of a snake can be modelled by a Serpenoid wave, which is a sinusoidal wave that propagates down the length of the snake's body.

To utilize the Serpenoid wave on the Carnagie Mellon robot, shown in Figure 1, [1] has shown that the wave can be discretely approximated as two waves, one for each set of joints. The angle of each joint is given by the equation

$$\theta_i(t) = \begin{cases} B_e + A_e sin(\omega t + \frac{4\pi k}{N} i + \delta) & i \text{ even} \\ B_o + A_o sin(\omega t + \frac{4\pi k}{N} i) & i \text{ odd} \end{cases}, \tag{1}$$

where k determines the number of cycles in each wave, N is the number of modules, δ is the phase offset between the even and odd waves, ω is the frequency of the wave, A_e and A_o are the oscillation amplitude of the even and odd joints angles respectively, and B_e and B_o are the dc offsets of the even and odd joint angles respectively. Gaits can then be designed by changing parameters in the wave [2].

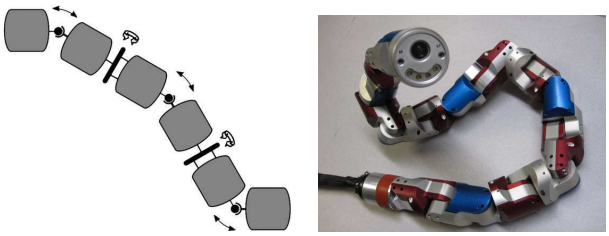

Fig. 1. The left figure shows a schematic of the snake robot design. Each joint axis is orthogonal to its neighbors and has a range of $\pm 90°$. The right image shows a picture of the snake created by the Biorobotics Laboratory at Carnegie Mellon University that will be used as the experimental platform [2], courtesy of Howie Choset.

III. Decentralized Implementation of the Serpenoid Wave

Despite the ease of using a sinusoid generator to implement (1), as previously done [1] [2], it is undesirable for gait transitions as it inherently depends on time. This dependence is detrimental because when the snake changes gaits it must try and coordinate a transition to another sinusoid wave which may have both a different value and direction. A naive approach to transition between sinusoid waves would be to blend the sinusoid waves over a period of time in the following manner

$$\theta_i(t) = (1 - g(t))\theta_{i1}(t) + g(t)\theta_{i2}(t) \tag{2}$$

where θ_{ij} denotes the desired angle value of the i^{th} joint executing the j^{th} gait and $g(t)$ is some weight function which transitions between 0 and 1 (for example a sigmoid function).

However, as shown in Figure 2, during transition the structure of the wave does not even appear sinusoidal. Moreover, when switching between gaits, not only does the module need to take into account its own sinusoid wave, but it must also account for the transition to a new phase offset between it angle and that of its neighbors.

As such, through a decentralized control scheme, we have created a framework to allow for a smooth transition between two different Serpenoid waves with different parameters. This framework achieves (1) without dependence on a time. Moreover, module i depends only on itself and module $i-1$ which will allow the modules to transition between gaits without communicating to a central control. To realize this framework we introduce a formulation for (1) based on limit cycles and give the necessary dynamics to execute a transition from one gait to another.

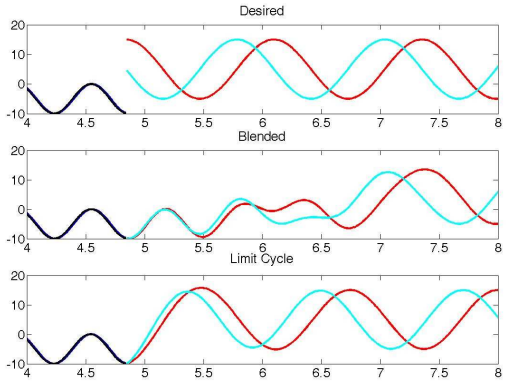

Fig. 2. The top figure shows the desired sinusoid waves, the middle shows the result of using a sigmoid function as a weight to blend waves during transition and the bottom shows using the limit cycle method. The black and red waves show the leader for the first and second gaits respectively. The blue and turquoise waves are the follower.

A. A Limit Cycle Approach

To achieve the desired properties, we have drawn inspiration from the area of Central Pattern Generators (CPG) where there has been an extensive amount of work on the design of cyclic motion (e.g. [6]). In particular, we build on a method presented in [8] which gives dynamics that can be written as the harmonic oscillator with added terms to allow for stable limit cycles with arbitrary frequency and radius.

We generalize the approach in [8] in two steps to allow for implementation of (1). First, we modify it to allow for arbitrary DC offset. We call the resulting dynamics f_{lead} (ie dynamics for an agent which does not depend on any neighbor). We then add a term which will allow for the modules to achieve a desired phase offset from their adjacent modules. We call the resulting dynamics f_{foll} (ie dynamics for an agent which is "following" another agent at a give phase offset). We can write both sets of dynamics as

$$f_{lead}(x_i) = \begin{bmatrix} \gamma_i & -\omega \\ \omega & \gamma_i \end{bmatrix} \hat{x}_i \tag{3}$$

$$f_{foll}(x_i, x_{i-1}) = f_{lead}(x_i) + c_2(x_{d_i} - x_i) \tag{4}$$

where

$$x_{d_i} = R(\phi_{d_i}) \frac{A_i}{A_{i-1}} \left(x_{i-1} + \begin{bmatrix} B_i - B_{i-1} \\ 0 \end{bmatrix} \right),$$

$\hat{x}_i = x_i - \begin{bmatrix} B_i & 0 \end{bmatrix}^T$, A_i and B_i are the desired amplitude and offset of module i, $R(\phi_{d_i})$ is a rotation matrix with angle ϕ_{d_i}, ϕ_{d_i} is the desired phase offset between modules i and $i-1$, ω is the frequency of oscillation,

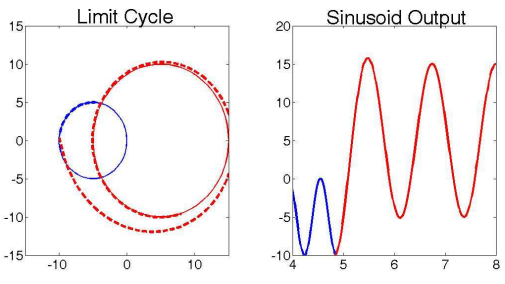

Fig. 3. This shows the correlation between a transition between different limit cycles using (3) and the corresponding sinusoidal output. On the left, the state switches from orbiting the smaller limit cycle to the large limit cycle. On the right is shown the resulting sinusoid wave.

$\gamma_i = c_1\left(\frac{A_i}{||\dot{x}_i||} - 1\right)$, c_1 is a gain proportional to the rate of convergence to the limit cycle, and c_2 is a weight proportional to the convergence rate of module i to achieve ϕ_{d_i}. The joint angle for module i would then be given by $\theta_i(t) = \begin{bmatrix}1 & 0\end{bmatrix} x_i(t)$.

All these parameters can be taken directly from (1) except for the weights, c_1 and c_2, and the desired phase offset, ϕ_{d_i}. The weights must be tuned to achieve desired convergence characteristics, which depend on the capacities of the motors at hand. We define ϕ_{d_i} as the desired offset between module i and module $i-1$. Observing (1), the value for ϕ_{d_i} can then be given by

$$\phi_{d_i} = \begin{cases} \frac{4\pi k}{N} + \delta & i \text{ even} \\ \frac{4\pi k}{N} - \delta & i \text{ odd} \end{cases}. \tag{5}$$

An example of the limit cycle approach implementing a gait transition can be seen in Figure 3, which graphically depicts the limit cycle transition for a single agent, and Figure 2, which shows the result of the cyclic persuit term for achieving desired phase offset.

B. Gait Transitions

We now set up the structure of the dynamics which will allow us to successfully implement gait transitions using (3) and (4). As the gait transition propagates down the snake, part of the snake will be executing the first gait and part of the snake will be executing the second gait. Therefore, when module $i-1$ switches to the second gait, module i must become the leader of the first gait (i.e. module i switches from $f_{foll,1}$ to $f_{lead,1}$). Otherwise, module i would be performing cyclic pursuit with a neighbor that is executing a different gait with different parameters for frequency and phase offsets.

With this in mind, we write the structure of the transition as

$$\dot{x}_1 = \begin{cases} f_{12}(x_1) & t < \tau_1 \\ f_{13}(x_1) & t \geq \tau_1 \end{cases} \tag{6}$$

$$\dot{x}_i = \begin{cases} f_{i1}(x_i, x_{i-1}) & t < \tau_{i-1} \\ f_{i2}(x_i, x_{i-1}) & \tau_{i-1} \leq t < \tau_i \\ f_{i3}(x_i, x_{i-1}) & t \geq \tau_i \end{cases}, \tag{7}$$

where τ_i denotes the time at which module i will switch from gate 1 to gate 2. We have replaced the subscripts "$foll$" and "$lead$" in (3) and (4) for ease of notation in Section IV. However, it should be noted that f_{i1} corresponds to agent i executing f_{foll} of the first gate, f_{i2} corresponds to f_{lead} of the first gate, and f_{i3} corresponds to f_{foll} of the second gate for $i > 1$ and f_{lead} of the second gate for $i = 1$.

C. Convergence

For (3) and (4) to be a valid implementation of the Serpenoid wave, we need to show that each module will converge to the desired limit cycle and phase offset from its leader agent.

Theorem 3.1: For sufficiently large c_2, $c_1 > 0$, $\dot{x}_1 = f_{lead}(x_i)$, and $\dot{x}_i = f_{foll}(x_i, x_{i-1})$ for $i = 2, 3, ..., N$, each module will converge to a stable limit cycle with radius A_i, offset B_i, and $x_i = x_{d_i}$ (x_{d_i} defined as in (4)).

Proof It can be verified that (3) is asymptotically stable to the desired limit cycle by transforming it into polar coordinates and evaluating the resulting linear system. As x_1 executes (3), it will converge to its desired limit cycle.

The proof of convergence for the remainder of the joint angles can be done in two steps. First, defining $e_i = x_{d_i} - x_i$, it can be shown that $V_{i1} = \frac{1}{2}||e_i||^2$ is a Lyapunov function as long as $||x_i||$ has some lower bound and c_2 is sufficiently large. It can then be shown that $V_{i2} = \frac{1}{2}||x_i||^2$ is always increasing for some ϵ and δ s.t. $\epsilon \leq ||x_i|| \leq \delta$ for sufficiently large c_1. This shows that $||x_i||$ does indeed have a lower bound. Details can be found in [9]. ■

IV. Decentralized Switch-Time Optimization

Smooth transitions are not enough to ensure a lack of unwanted movement; the snake must maintain certain properties to maintain the structure of the gait during transition. To accommodate these properties, we build upon current switch time optimization techniques (eg [7]) to allow each module to compute the optimal time, given some cost, to switch between two gaits in a decentralized manner. In this section, we first set up the switch-time optimization problem using Dual Decomposition which allows a cost to be minimized in a decentralized fashion [10]. We will then show the solution to the problem using a technique called Uzawa's Algorithm [11].

A. Dual Decomposition

Dual Decomposition is an emerging tool for optimizing costs in multi-agent systems (eg [12], [13]). It allows each agent to maintain its own version of the variables that it needs for optimization and ensures the equality with its neighbors' versions through the introduction of Lagrange multipliers. We use the notation τ_{ij} denoting module i's version of τ_j and similarly x_{ij} denoting module i's version of x_j.

We denote the cost assigned to module i as J_i and let it take the form

$$J_i(\bar{x}_i) = \int_{\tau_0}^{\tau_{N+1}} C_i(\bar{x}_i(t))dt \tag{8}$$

where $C_i(\bar{x}_i)$ is the instantaneous cost associated with module i which depends on itself and its neighbors, $\bar{x}_i = [x_{ij}^T]\ \forall j \in I_i$, $I_i = \{j|\frac{\partial C_i}{\partial x_j} \neq 0$ or $\frac{\partial f_{ik}}{\partial x_j} \neq 0$ for $k = 1,2,3\}$, τ_0 and τ_{N+1} correspond to the initial and final times respectively, and τ_i, $i = 1,...,N$, corresponds to the time at which agent i switches to gait 2.

Using this notation, where we have absorbed all decision variables into μ and τ, we can write the Dual Decomposition problem as

$$\max_{\mu}\min_{\tau} J = \sum_{i=1}^{N} J_i(\bar{x}_i) + \sum_{i=1}^{N}\sum_{j\in I_i} \mu_{ij}(\tau_{ij} - \tau_{jj}) \tag{9}$$

$$\text{s.t. } \dot{x}_{ik} = \begin{cases} f_{k1}(x_{i,k}, x_{i,k-1}) & t < \tau_{i,k-1} \\ f_{k2}(x_{i,k}, x_{i,k-1}) & \tau_{i,k-1} \leq t < \tau_{i,i} \\ f_{k2}(x_{i,k}, x_{i,k-1}) & t \geq \tau_{i,k} \end{cases},$$

$$x_{ik}(\tau_0) = x_k(\tau_0)$$

$$\forall k \in I_i, i = 1,...,N$$

However, we still have one problem dealing with separability of costs as required by Dual Decomposition. Despite the fact that we have said that the instantaneous cost, C_i, only depends on a local neighborhood, this does not imply that J_i depends on a local neighborhood. Allowing $k_i = \max_j s.t. j \in I_i$, we will show that $\frac{\partial J_i}{\partial \tau_q} = 0$ for $q > k_i$. However, we will also show that for $q < k_i$, $\frac{\partial J_i}{\partial \tau_q}$ is not necessarily zero. This means that J_i depends on all x_j, $j \leq k_i$.

Therefore, to properly use Dual Decomposition, module i must maintain versions of x_j and τ_j $\forall j \leq i$, but this is not decentralized as module N would then require global information. One way around this is to let module i assume that module $i-1$ is a leader agent and, as such, it only maintains values of the variables indexed by I_i. This is mathematically equivalent to assuming that $\frac{\partial J_i}{\partial \tau_q} = 0\ \forall j \notin I_i$. This has produced good results, as shown in Section V.

B. Uzawa's Algorithm

Now that we have broken down the problem using Dual Decomposition, it is in a form where we can propose a solution. Uzawa's algorithm allows for solving both the min and max simultaneously by taking a step in the gradient descent direction over the variables being minimized and a step in the gradient ascent direction over the variables being maximized [11]. In other words, (9) can be solved iteratively by following the following strategy

$$\tau_{mn}^{k+1} = \tau_{mn}^{k} - \eta\frac{\partial J}{\partial \tau_{mn}} \tag{10}$$

$$\mu_{ql}^{k+1} = \mu_{ql}^{k} + \eta\frac{\partial J}{\partial \mu_{ql}},$$

where the superscript denotes the iteration number and η is a gradient step size.

This allows for a multi-agent system to solve the Dual Decomposition problem given in (9). Module i can use (10) to update its values of $\tau_{ij}, j \in I_i$. It can then communicate τ_{ij} to, as well as receive τ_{jj} from, module j. In this fashion each agent can then update its value of the Lagrange multiplier μ_{ij}. For modules to calculate the solution, we give an explicit form for the gradients which can be evaluated using Theorem 4.1.

$$\frac{\partial J}{\partial \tau_{mn}} = \frac{\partial J_m}{\partial \tau_{mn}} + \mu_{mn},\ m \neq n \tag{11}$$

$$\frac{\partial J}{\partial \tau_{mm}} = \frac{\partial J_m}{\partial \tau_{mm}} - \sum_{\{j:m\in I_j\}} \mu_{jm}$$

$$\frac{\partial J}{\partial \mu_{ql}} = \tau_{ql} - \tau_{ll}$$

Theorem 4.1: With the dynamic structure of a directed line graph, the gradient for $\frac{\partial J_m}{\partial \tau_{mn}}$ found in (11) can be written as

$$\frac{\partial J_i}{\partial \tau_{ij}} = \big(\lambda_{ij}\big(f_{j2} - f_{j3}\big) + \lambda_{i,j+1}\big(f_{j+1,1} - f_{j+1,2}\big)\big)|_{\tau_{ij}} \tag{12}$$

where, allowing $k_i = \max_j\{j|j \in I_i\}$,

$$\dot{\lambda}_{im} = -\big(\frac{\partial C_i}{\partial x_m} + \lambda_{im}\frac{\partial f_{ml_m}}{\partial x_m} + \lambda_{i,m+1}\frac{\partial f_{m+1,l_{m+1}}}{\partial x_m}\big),$$

$$\lambda_{im}(\tau_{N+1}) = 0,\ \forall m \in \{j|j \in I_i \text{ and } j \neq k_i\}$$

$$\dot{\lambda}_{ik_i} = -\big(\frac{\partial C_i}{\partial x_{k_i}} + \lambda_{ik_i}\frac{\partial f_{k_i l_{k_i}}}{\partial x_{k_i}}\big), \lambda_{ik_i}(\tau_{N+1}) = 0$$

and

$$l_j = \begin{cases} 1 & \tau_0 \leq t < \tau_{j-1} \\ 2 & \tau_{j-1} \leq t < \tau_j \\ 3 & \tau_j \leq t \end{cases}$$

Fig. 4. Snapshots of the snake during transition are shown while time increases from left to right. The propagation of the switch from Straight to Sidewind can be observed as the part of the snake that remains straight is still executing the first gait.

Proof Using the cost in (8)and dynamics for the state as defined in (9) we can augment the cost with the dynamics as

$$\hat{J}_i = \sum_{j=0}^{N} \int_{\tau_j}^{\tau_{j+1}} \Big(C_i(X_i) + \sum_{k=1}^{N} \lambda_{ik}(f_{kl_k} - \dot{x}_{ik}) \Big) dt$$

Using standard variational arguments we vary $\tau_{ij} \rightarrow \tau_{ij} + \epsilon v_{ij}$ which causes the state to vary as $x_{ik} \rightarrow x_{ik} + \epsilon \eta_{ik}$. We can write

$$\begin{aligned}
&\frac{1}{\epsilon}\big(\hat{J}_i(\tau - \epsilon v) - \hat{J}_i(\tau)\big) = \\
&= \int_{\tau_0}^{\tau_{N+1}} \sum_{m=1}^{N-1} \Big(\frac{\partial C_i}{\partial x_m} + \lambda_{im} \frac{\partial f_{ml_m}}{\partial x_m} + \\
&\lambda_{im+1} \frac{\partial f_{m+1 l_{m+1}}}{\partial x_m} + \dot{\lambda}_{im} \Big) \eta_{im} dt \\
&+ \int_{\tau_0}^{\tau_{N+1}} \Big(\frac{\partial C_i}{\partial x_N} + \lambda_{iN} \frac{\partial f_{Nl_N}}{\partial x_N} \Big) \eta_{iN} dt \\
&+ \sum_{j=1}^{N-1} v_{ij} \Big((\lambda_{ij}(f_{j2} - f_{j3}) + \\
&+ \lambda_{i,j+1}(f_{j+1,1} - f_{j+1,2}) \Big) |_{\tau_{ij}} \\
&- \sum_{k=1}^{N} \lambda_{ik} \eta_{ik} |_{\tau_{N+1}} + o(\epsilon)
\end{aligned}$$

So we allow the costate to be defined as

$$\dot{\lambda}_{im} = -\Big(\frac{\partial C_i}{\partial x_m} + \lambda_{im} \frac{\partial f_{ml_m}}{\partial x_m} + \lambda_{i,m+1} \frac{\partial f_{m+1,l_{m+1}}}{\partial x_m} \Big),$$

$$\lambda_{im}(\tau_{N+1}) = 0, m = 1, ..., N-1$$

$$\dot{\lambda}_{iN} = -\Big(\frac{\partial C_i}{\partial x_N} + \lambda_{iN} \frac{\partial f_{Nl_N}}{\partial x_N} \Big), \lambda_{iN}(\tau_{N+1}) = 0$$

and we obtain the gradient in (12).

Furthermore we can show through induction that for $q > k_i$, $\lambda_{iq} = 0 \ \forall t$ (and thus $\frac{\partial J_i}{\partial \tau_{im}} = 0$). First we examine λ_{iN} which is straight forward to see that it is always zero if $\frac{\partial C_i}{\partial x_N} = 0$ using properties of the state transition matrix (eg [14]). Using the same argument, it is easy to show that for all $q > k_i$ $\lambda_{iq} = 0$ if $\lambda_{i,q+1} = 0$. This leaves us with the resulting costate equations given in (12). ■

V. Transition Example

As mentioned throughout this paper, it is important to maintain properties of the gait during transition. One such property that has been shown to be important is the phase difference between adjacent modules [1]. The implementation of the Serpenoid wave using limit cycles easily allows for the calculation of the phase between two modules. It is noted that the inner product between two vectors can be calculated as $x_i x_{i-1}^T = ||x_i|| ||x_{i-1}|| \cos(\psi)$ where ψ is the angle between the two vectors. In our limit cycle approach, $\psi = |\phi_{d_i}|$, where ϕ_{d_i} is the desired phase between the angle of modules i and $i-1$. Using this information, we design the instantaneous cost to punish the difference between desired and actual phase difference:

$$C_i(x_i, x_{i-1}) = k\Big(\frac{x_i^T x_{i-1}^T}{||x_i|| ||x_{i-1}||} - cos(\phi_{di}) \Big)^2 \qquad (13)$$

Straight To Sidewind

A transition from a Straight orientation a Sidewind gait provides a good example for the utility of performing the switch time optimization as the snake often begins in a Straight orientation. Both gaits are straight forward to implement. The Straight gait can be implemented using a proportional control to move the angles to zero (ie x value where $\begin{bmatrix} 1 & 0 \end{bmatrix} x = 0$ for the limit cycle approach). A sidewind gait is a typical gait used by a snake to move quickly sideways [2], [3]. It is designed by using the Serpenoid wave with the phase offset between

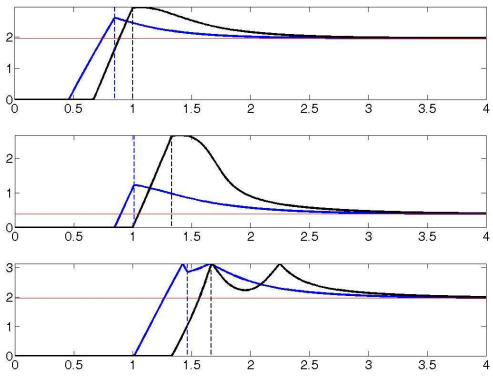

Fig. 5. This figure shows the results of the transition between a Straight gait to Sidewind gait. It shows the initial guess vs the optimized values for the phase difference in black and blue respectively of three of the modules of the snake. The vertical lines show when the switch time occurred.

odd and even modules, δ, equal to $\frac{pi}{4}$ and $B_o = B_e = 0$ [2].

The solution is somewhat intuitive, ie a module should not start switching until the module in front of it is close to the correct phase offset. This will leave the unswitched portion of the snake in a Straight configuration while the switching propagates down the snake, as shown in Figure 4. Figure 5 shows a comparison of the performance of three of the sixteen modules with optimal switch times with respect to arbitrarily chosen switch times. As can be seen, the optimized version has a much smaller deviation from the desired phase after the switch time than does the unoptimized version.

VI. Conclusion

We have presented a method for decentralized switch time optimization for gait transitions for a snake robot. We have done this by presenting both a framework for gaits which will allow for smooth transitions and theory for optimal switch-times which will allow the snake to determine the optimal time for each module to switch to a new gait.

We demonstrated the utility of this method through an example where it was evident that the switch-time optimization was able to reach the desired phase difference with less oscillation. To do so, the transition propagated down the snake instead of having each module switch at the same time. This provides for less undesired oscillations during transition between gaits and expect that this will make the movement of the snake during transition more predictable, as this will be an area of future work.

Acknowledgements

The authors are grateful to Howie Choset for helpful comments and discussions. This work was sponsored by the DARPA M3 program.

References

[1] J. Gonzalez-Gomez, "Modular robotics and locomotion: Application to limbless robot," Ph.D. dissertation, Universidad Autonoma de Madrid, 2008.

[2] M. Tesch, K. Lipkin, I. Brown, R. Hatton, A. Peck, J. Rembisz, and H. Choset, "Parameterized and scripted gaits for modular snake robots," *Advanced Robotics*, vol. 23, no. 9, pp. 1131–1158, 2009.

[3] J. Burdick, J. Radford, and G. Chirikjian, "A 'sidewinding' locomotion gait for hyper-redundant robots," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, may 1993, pp. 101 –106 vol.3.

[4] Y. Umetani and S. Hirose, "Biomechanical study of serpentine locomotion," in *Proc. 1st RoManSy Symp*, vol. 73, 1974, pp. 171–184.

[5] S. M. LaVell, *Planning Algorithms*. Cambridge: Cambridge University Press, 2006.

[6] A. Jan and Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642 – 653, 2008.

[7] M. Egerstedt, Y. Wardi, and F. Delmotte, "Optimal control of switching times in switched dynamical systems," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 3, dec. 2003, pp. 2138 – 2143 Vol.3.

[8] J. Buchli and A. Ijspeert, "Distributed central pattern generator model for robotics application based on phase sensitivity analysis," *Biologically Inspired Approaches to Advanced Information Technology*, pp. 333–349, 2004.

[9] G. Droge and M. Egerstedt, "Optimal decentralized gait transitions for snake robots," School of Electrical and Computer Engineering, Georgia Institute of Technology, http://gritslab.gatech.edu/home/wp-content/uploads/2012/01/Dist_Gaits.pdf, Tech. Rep., January 2012.

[10] S. Boyd, *Convex Optimization*. Cambridge: Cambridge University Press, 2004.

[11] K. Arrow, L. Hurwicz, and H. Uzawa, *Studies in Nonlinear Programming*. Stanford University Press, Stanford, CA, 1958.

[12] P. Giselsson and A. Rantzer, "Distributed model predictive control with suboptimality and stability guarantees," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 7272–7277.

[13] P. Twu, R. Chipalkatty, A. Rahmani, M. Egerstedt, and R. Young, "Air traffic maximization for the terminal phase of flight under faa's nextgen framework," in *Digital Avionics Systems Conference (DASC), 2010 IEEE/AIAA 29th*, oct. 2010, pp. 2.C.1–1 – 2.C.1–14.

[14] W. L. Brogan, *Modern Control Theory*. New York: Quantum Publishers, 1974.