A Self-Organization Strategy for Unmanned Autonomous Systems

Benjamin Lee[1], Sehwan Oh[2], Michael Balchanos[3], George Vachtsevanos[4]

[1,2,3,4]*Georgia Institute of Technology, Atlanta, GA, 30024, United States of America*

blee78@gatech.edu
soh48@gatech.edu
michael.balchanos@ asdl.gatech.edu
gjv@ece.gatech.edu

ABSTRACT

Complex systems are constructed from multiple subsystems and components with each serving incremental tasks, where the "emergent" system behavior cannot be deduced from the behaviors of the individual parts. The key requirement of complex systems is the ability to compensate for unforeseen and extreme disturbances, so it is important to design a control method that ensures acceptable level of system resilience throughout its operation. Therefore, detailed and accurate knowledge of system behaviors is paramount for the design of complex system control strategies. This paper presents a self-organizing control strategy that incorporates both situational awareness and failure impact compensation for a resilient unmanned autonomous system.

1. INTRODUCTION

Complex systems are "systems of systems" comprised of hierarchical sets of subsystems or components, where the combined simultaneous operation of many components can lead to unforeseen "emergent" behaviors. Vinerbi et al. (2010) suggest that even though good knowledge of system behaviors is significant, "full" knowledge of complex system behavior may not be achievable. Also, complex systems are vulnerable to multiple failures at once, while the effect of individual failures may not be evident. Complexity Theory has shown to improve understanding of system behavioral modes and provide a viable means for modeling of such complex systems. This paper postpones discussion of Complexity Theory attributes to a future document and focuses on the resilient design via self-organization.

Unexpected change in complex system behavior occurs in case of extreme disturbances, such as a complete failure of a subsystem. The most commonly used robust control technique for disturbance rejection is PID (feedback) control. A major limitation of traditional control systems using PID is the lack of online adjustments to changing system properties. Manual computation and adjustment to new system behaviors are impractical for complex system applications such as unmanned autonomous systems (UAS).

A self-organizational method could be an alternative to the traditional robust control avoiding a heavy computational burden. A system is considered *organized* if it has certain structure and functionality, and self-organization implies that the organization of the system occurs internally, without any external or centralized control unit (Prehofer and Bettstetter, 2005). In the simplest case, a self-organization strategy consists of two components: response and adaptation, responding to the system's functionality. Therefore, systems with the same structure may require a different adaptation strategy depending on the system's operational objectives. Along with a reduced computational burden due to the targeted operation, a self-organization method provides the benefit of random noise adaptation, since the process is spontaneous with intrinsic update rules (Heylighen, 1999).

In this paper, a novel self-organizational method is introduced as a compensatory measure to maintain system functionality under the presence of failure modes. It is noted that resilience requirements refer to severe disturbances, i.e. failure modes compared to usual disturbances compensated by conventional technologies such as robust or PID control. A typical unmanned autonomous ground vehicle – the hexapod – is employed as the testbed for the development and validation of the self-organizing strategy. Methods to understand system behavior include data acquisition, system modeling, and proper construction of performance metrics; the strategy includes a policy to address the changing system conditions and success criteria to evaluate the optimal action. The physical, functional, nonlinear dynamic, and graph theoretic models will be considered to examine system behaviors under both normal and faulty conditions. Then, the self-organization strategy will be introduced in the form of a Markov Decision Process (MDP) with dynamic programming for optimal performance. Finally, the success criteria for the control method will be constructed with Lyapunov stability conditions so that the self-organization strategy can be modified throughout the system operation for system resilience regarding stability and resource limitations. Simulation results will be presented at the end to demonstrate the efficacy of the approach.

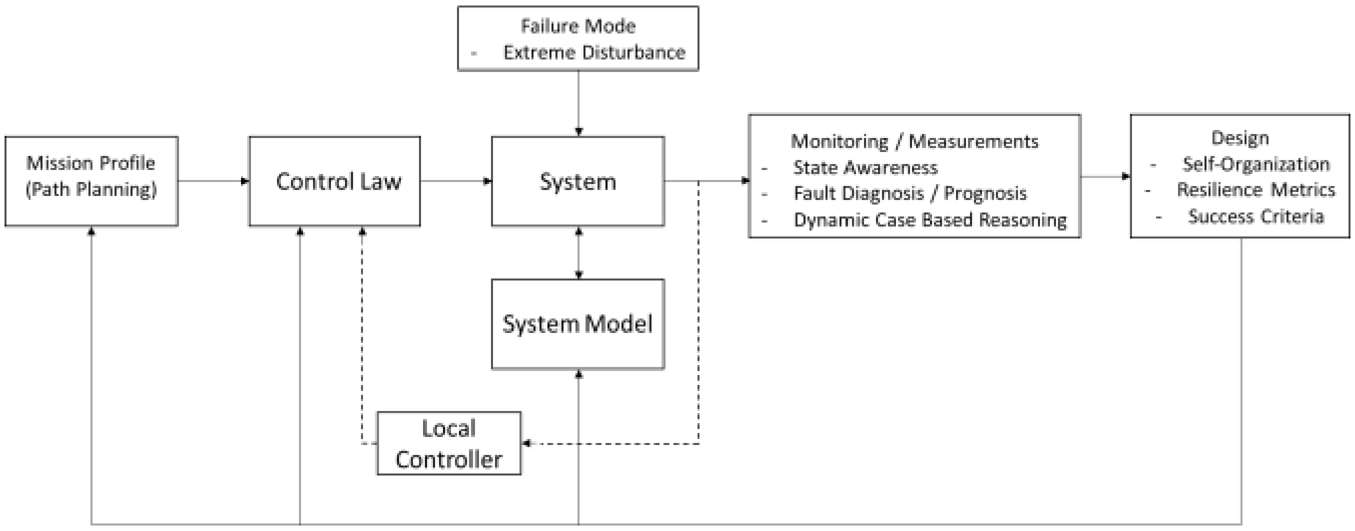

Figure 1. Overall self-organization methodology.

2. TECHNICAL APPROACH

2.1. The Methodology: An Overview

Figure 1 depicts a schematic of the framework for self-organization of a complex system subjected to severe disturbances, namely failure modes. Initially, a system-level mission profile is suggested, where the system is commanded to travel a prescribed path in 2D or 3D spaces. For traditional disturbance rejection, the system will operate under a control law, which employs local controllers for feedback compensation. In the presence of a failure mode, the mission is unachievable, and the proposed self-organizational method takes over as the higher-level control architecture.

The self-organization strategy begins with two types of modeling techniques aimed to describe the system behaviors in data format; a graph theoretic model to examine the current state of the system, and a dynamic/kinematic model to predict the optimal next step. Along with the data obtained from the models, monitoring processes such as situation awareness and fault diagnosis/prognosis will examine the current health state of the system, and the optimal action will be determined based on a fully implemented MDP. The MDP output will be the optimal action for the system to proceed to the next step of the mission profile while maintaining an acceptable level of stability and observing physical/functional constraints. The system model and control law will be updated based on the evaluation of the outcome of the process through appropriate success criteria.

2.2. Self-Organization Strategy

2.2.1. Spectral Graph Theory

A graph G is a set $G(V,E)$ that consists of vertices and edges (connections between vertices). In systems engineering, the structure of a complex system can be represented as a graph with the list of system nodes and their respective interconnectivity.

Wilson (1996) lists several useful matrix representations in graph theory. The adjacency matrix A is a $n\times n$ matrix where the element a_{ij} indicates connectivity from node i to node j with 1, and 0, otherwise. The degree matrix D is a $n\times n$ diagonal matrix, where the diagonal elements d_i are the degree (number of edges) of node i.

The Laplacian matrix L is defined as in Equation (1) and is also called the system matrix.

$$L = D - A \tag{1}$$

The second smallest eigenvalue (denoted by λ_2) of the Laplacian matrix is called the algebraic connectivity of the graph and represents how well the graph is connected. The algebraic connectivity is also an important indication of a network's resilience, and the quantification of the importance of a node or a link with the effect of node removal on the algebraic connectivity is studied in the work of Liu et al. (2009).

Another useful matrix in spectral graph theory is the transition matrix T, in which the element p_{ij} is the probability of transitioning from node i to node j. The aforementioned adjacency and degree matrices can be combined as in Equation (2) to determine the transition matrix within a system graph (Butler, 2008):

$$T = D^{-1}A \tag{2}$$

The transition matrix is a versatile tool to represent probabilistic processes. In addition to storing the probabilities within nodes of a system, the transition matrix can also represent state-transition probabilities.

2.2.2. Markov Decision Process

Proper definition of system states and the state transition probabilities is the first step for constructing a Markov Decision Process (MDP), formulated by Bellman (1957), and applying the Markovian property to move from the current state to the next optimal state based on a predefined policy. MDP is a dynamic programming method, where the control problem for a complex system is divided into simpler sub problems in a memory based structure so that the solution for the next occurring sub problem can be looked up immediately instead of re-composing the solution, thereby reducing significantly the computational burden compared to traditional control methods.

MDP is a set of decision-making rules consisting of {S,A,T,R}, where S is the finite set of achievable states, A is the finite set of actions that connect a state to other states, T is the transition matrix that stores the likelihoods of the state transitions, and R is the reward matrix that indicates the immediate effect of an action applied to a state. In other words, the transition matrix can be used to describe the system's behavior, and the reward matrix can be used to guide the control action towards mission completion. Yukalov and Sornette (2014) suggest that any complex system, under given conditions, is more inclined to occupy the most stable state. From the work of Gabbai (2005), self-organization is an evolving process towards a state of equilibrium, commonly called an attractor. An attractor could provide a lower dimensional representation of complex system dynamics, and an example attractor could be a desired path for the system to follow. Therefore, the transition and reward matrices should be constructed with higher probability and reward assigned for approaching the desired path.

2.2.3. Dynamic Programming

The main goal of MDP is to construct a "policy" $\pi(s)$ that provides the optimal available action for each state. The algorithm to obtain the policy is represented by Equation (3), known as the Bellman equation. Value functions, V(s), are defined for each state to accumulate the immediate reward from an action at each time step, until the overall reward converges to steady values.

$$V(s) = \max(R(s,a) + \gamma \textstyle\sum_{s'} T(s'|s,a)V(s')) \qquad (3)$$

The discount factor $\gamma \in [0,1]$ is used to suppress future rewards and ensure convergence of the overall reward. Based on the solution of the Bellman equation, the policy will return the optimal action for a state with the maximum overall reward.

The policy obtained as the output of the MDP is constantly updated through success criteria for desirable system performance, while the system is supporting a self-organization control method that spontaneously and internally compensates for severe disturbances.

2.3. Self-Organization Method for a Hexapod

To illustrate the proposed self-organization method, a hexapod robot is selected as the test system. The mission profile is set for the hexapod to travel from a current point A to a goal point B in a straight-line path. Cully et al. (2015) suggest an improved trial and error method to determine the optimal action for a walking hexapod with a broken leg, but the large original search space and minutes of lengthy adaptation time to the next step hamper their development. Instead, a self-organization method that spontaneously generates the optimal action can provide an alternative to decrease significantly the computational burden.

2.3.1. Hexapod Dynamic/Kinematic Model

The hexapod used in this case study is composed of a body and six legs. Each leg is a three DOF (degrees of freedom) subsystem with three servo joints. From the work of Sorin et al. (2011), the names and functions of each joint and link are shown in Figure 2.

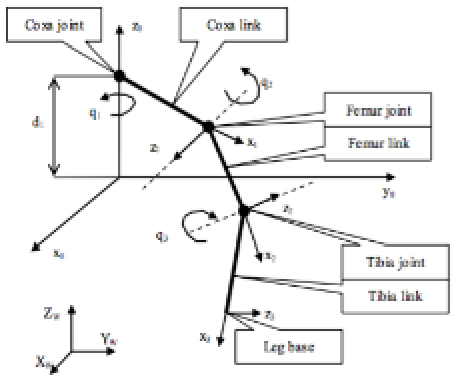

Figure 2. Hexapod leg structure

It can be seen that the Coxa joint rotates horizontally, and the other joints rotate vertically. The hexapod leg's kinematic movement is governed by the joint motor rotations, and the position of the end point of a leg contacting the ground with respect to the hexapod body can be obtained by applying consecutive coordinate transformation matrices, assuming the origin as the center of the body, as shown in Equations (4,5,6,7) (Barai et al. 2013). In the equations, ${}^{i}_{j}T$ is the transformation matrix from joint i to joint j, s_i and c_i are the sine and cosine functions of ith joint angle, L_i is the link length, and P_f is the end position of the leg.

$$ {}^{0}_{3}T = {}^{0}_{1}T * {}^{1}_{2}T * {}^{2}_{3}T \tag{4}$$

$$ {}^{0}_{3}T = \begin{bmatrix} c_1(c_2c_3 - s_2s_3) & c_1(-c_2s_3 - s_2c_3) & -s_1 & c_1c_2L_1 \\ s_1(c_2c_3 - s_2s_3) & s_1(-c_2s_3 - s_2c_3) & c_1 & s_1c_2L_1 \\ -c_2s_3 - s_2c_3 & s_2s_3 - c_2c_3 & 0 & -s_2L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$ {}^{0}P_f = {}^{0}_{3}T\,{}^{3}P_f = {}^{0}_{3}T \begin{bmatrix} L_2 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} P_{fx} \\ P_{fy} \\ P_{fz} \\ 1 \end{bmatrix} \tag{6}$$

$$ \begin{aligned} P_{fx} &= L_2c_1(c_2c_3 - s_2s_3) + c_1c_2L_1, \\ P_{fy} &= L_2s_1(c_2c_3 - s_2s_3) + s_1c_2L_1, \\ P_{fz} &= -L_2(c_2s_3 + s_2c_3) - s_2L_1. \end{aligned} \tag{7}$$

All hexapod joints have a range of reachable angles ($\in [\theta_{min}, \theta_{max}]$), and the combination of the horizontal rotation of the Coxa joint and the vertical rotations of the Femur and Tibia joints create a set of reachable workspaces for the legs. The hexapod locomotion to move to the goal position will be assumed a conventional tripod gait. At each time step, three legs are swinging in space, and the other three are supporting the robot on the ground. The duty factor of the gait between the left and right side of the hexapod is 0.5, implying an equal duration in alternating sides, where the swinging legs at each step are the middle leg on one side and the front and rear legs on the other side. The walking direction of the hexapod can be determined by combining the direction vectors of the three swinging legs at each step. Assuming normal condition with no disturbance, the legs will swing forward in the direction of the desired path, and the three direction vectors from the swinging legs can be combined, as shown in Figure 3, where the arcs indicate the reachable workspaces of the swinging legs.

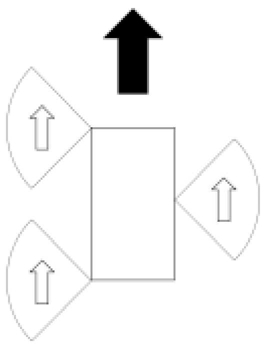

Figure 3. Upper view of the walking direction of the hexapod from the swinging motion

2.3.2. Failure Mode (Locked Joint Failure)

The three joints on each leg can be combined for the hexapod to have 18 DOF. Such high degree of manipulation allows versatile motion for the hexapod, but the complexity also induces vulnerability to severe failure modes. An example of a possible failure mode in a hexapod is the locked joint failure, where a joint angle is fixed at a certain state and cannot be controlled. In the work of Yang (2003), locked joint failures of different joints are shown to result in different effects on the leg workspace. Since the Coxa joint is in charge of the horizontal swinging movement of the leg, locked joint failure at the Coxa joint completely disables the leg's swinging motion and the leg can only lift and plant itself vertically. On the other hand, locked joint failure at the vertically operating Femur and Tibia joints will have no effect on the swinging motion, but will diminish the leg's stretchable length, so the upper view of the leg's workspace will have the same arc shape but with reduced size. The impact of locked joint failures on the leg's workspace will cause the hexapod to derail from its original path in an unexpected manner.

2.3.3. Hexapod Graph Model

Figure 4 shows a graph representation of a hexapod with each node numbered appropriately.

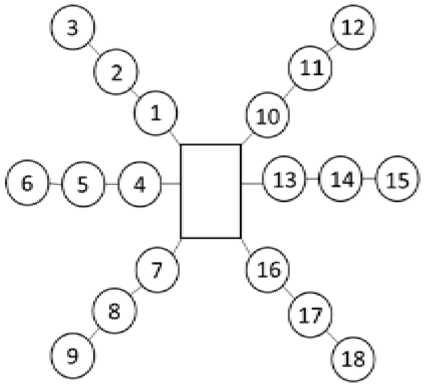

Figure 4. Graph representation of a hexapod with node numbers

The impact of a locked joint failure in a hexapod can be represented simply by a node removal. More specifically, a failed node loses controllability and converts into an edge, so the adjacency and degree of the corresponding node become zero. The aforementioned algebraic connectivity can be used to indicate the presence or severity of locked joint failure(s). Demonstration of the effect of locked joint failure on algebraic connectivity is shown in Table 1, where the algebraic connectivity of the hexapod system graph is evaluated in normal condition first followed by the failure mode at node 8, and finally another failure mode at node 15.

Table 1. Effect of failure mode on algebraic connectivity.

# of failure mode	Algebraic connectivity
0	0.3384
1	0.04874
2	-3.0199e-16 ≈ 0

It is shown that the incremental addition of failure modes decreases the algebraic connectivity, and when there are two failure modes present, the algebraic connectivity becomes zero, meaning the graph is disconnected.

2.3.4. Hexapod MDP

The MDP formulation is applied, as the self-organization strategy, to the hexapod under a locked joint failure. As suggested by Cuaya-Simbro and Munoz-Melendez (2008), the state space S can be defined as the valid positions of the legs, and the action space A as the transitions that enable the robot to move from one valid state to another. The finite deterministic case of the MDP algorithm is used for this problem, and the state space is represented by three available leg positions at each step (front, aligned, and rear with respect to the leg's connection to the body, as shown in Figure 5).

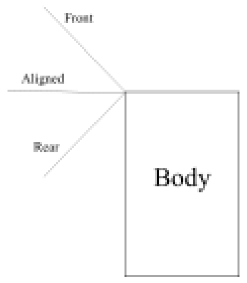

Figure 5. Example valid state space of a hexapod leg

Since three legs (two from one side and one from the other) either swing or support the system identically in a tripod gait, there are 9 total available states. Assuming the states can move to any other valid state, the action space A will have the same dimension as the state space.

The MDP solution in this problem will be a policy that maps the optimal action for each state for the hexapod to move along the desired path. Chades et al. (2014) provide a Markov Decision Process Toolbox for MATLAB to compute the MDP solution. By initializing the S×S×A transition and reward matrices, where S is the number of states, and A is the number of actions, the finite horizon solution of the MDP is concluded in N number of steps. In addition to the policy, the toolbox also outputs the used CPU time to solve the problem, which can be used to compare the computational burden to that of traditional robust control methods.

2.3.5. Success Criteria (Lyapunov Stability)

The hexapod's moving path following the MPD policy solution is evaluated via Lyapunov stability conditions to verify the stability and effectiveness of the self-organization method. Path-based Lyapunov stability analysis of a hexapod is detailed in the work of Jeong et al. (2013), where the positional error vector, e, is defined as in Equation (8) so that the Lyapunov function V and the Lyapunov equation are stated in Equation (9).

$$e = \begin{pmatrix} x_e \\ y_e \end{pmatrix} = \begin{pmatrix} x_d - x_c \\ y_d - y_c \end{pmatrix} \tag{8}$$

$$\begin{cases} V = e^T K e \\ A^T K A - K + Q = 0 \end{cases} \tag{9}$$

x_d denotes the desired position, x_c denotes the current position, A is the state transition matrix where $\overrightarrow{x_{t+1}} = A\overrightarrow{x_t}$, and the matrices K and Q are symmetric positive definite matrices.

3. RESULTS

Using the MDP Toolbox, the 9×9×9 transition and reward matrices can be defined following the hexapod structure and tripod gait behavior. Assuming 10 steps between current and goal points, the finite horizon MDP problem can be solved to give the value functions for each state, policy of optimal action for the states at each time step, and the used CPU time to compute the policy. The value functions of the first five states (S1 ~ S5) are shown in Figure 6, where the value (immediate reward) converges to zero over the time steps for each state so the overall reward converges to a certain value.

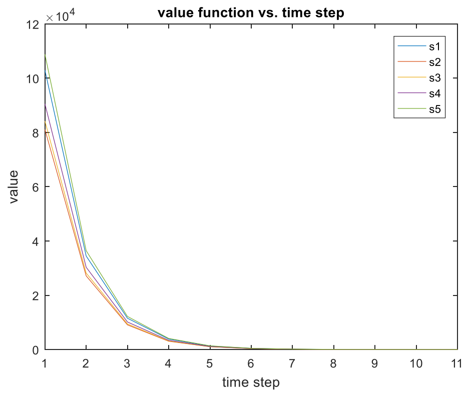

Figure 6. Value function computed over 10 time steps

The policy is generated as a 9×10 matrix where each row corresponds to a state and the columns store the optimal action for the state at each time step. A graph representation of the policy (first five states, for simplicity) is shown in Figure 7.

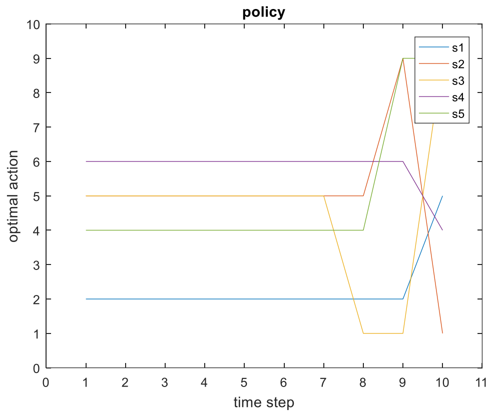

Figure 7. Policy generated from MDP Toolbox

The notable part of the MDP Toolbox demonstration is the CPU time, which results in 0.0156s for mission completion. System parameters need to be adjusted in case of a failure mode, as the change in dynamics can cause unexpected system behaviors. Since commonly used robust control methods with PID controllers are designed for systems with predefined dynamics and properties, spontaneous and reliable adjustment to new system dynamics with PID controllers is impractical in real life.

Considering the hexapod for example, the lengthy settling time required for the PID controllers to reach steady state will cause the hexapod to take several seconds to adjust to a different configuration while tuning the gains for all 18 joint motors. Even assuming perfectly synchronous joint rotations, using PID control for a real-time hexapod gait is impractical. Moreover, compared to the minutes of computation needed for the trial and error method mentioned in Section 2.3, the proposed self-organization method offers a holistic guide of system behavior for disturbance accommodation with dramatically less computational burden.

Assuming the hexapod starts at the origin, a diagonal path in the XY plane can be assumed to be the desired path for the hexapod to travel. A locked joint failure is added to the left-middle leg of the hexapod to test the self-organization behavior. For both conditions (with or without failure mode), the hexapod behavior is governed by the MDP policy. The resulting paths are shown in Figure 8 where the hexapod travels along the desired diagonal path. With a failure mode present, the hexapod abruptly moves to the left due to the failure mode and then gradually converges back to the desired path.

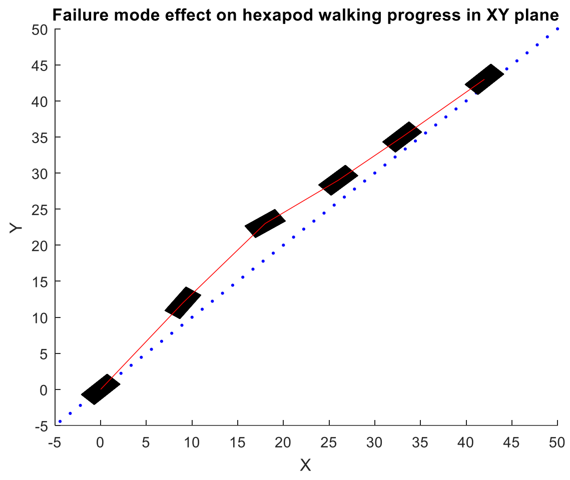

Figure 8. Hexapod walking path in nominal condition and with locked joint failure

To verify that the hexapod with locked joint failure returns to a stable mode, the distance from the desired path (= difference between x and y coordinates) can be used as the error value and apply the Lyapunov stability conditions. The stability condition is constructed as in Equation (10), assuming K = 1 or the identity matrix (for dimensions larger than 2).

$$\begin{cases} e_t = |y_t - x_t| \\ Q = 1 - A^2 = 1 - \left(\frac{e_{t+1}}{e_t}\right)^2 > 0 \end{cases} \tag{10}$$

In other words, the positional error of the vehicle must decrease in magnitude at each time step for the process to be stable. The resulting Q value evaluation is shown in Figure 9.

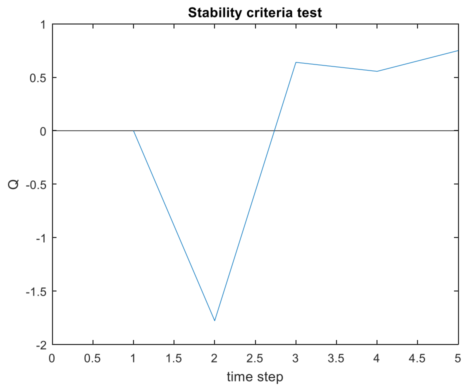

Figure 9. Q value of the hexapod with locked joint failure

It can be seen from Figure 9 that the process becomes unstable when the failure mode is initiated, then returns to a stable state between the second and third time steps. Upon detection of noticeable digression from the desired path, the predetermined MDP policy guides the hexapod to return to traveling in the direction of the desired path, which shows in the stability evaluation as the Q value drops to negative upon occurrence of a failure mode then turns positive as self-organization process is activated.

4. CONCLUSION

The self-organization method proposed in this paper combines the Markov Decision Process with Lyapunov stability conditions for a complex system to maintain stability under a severe failure mode. The proposed method demonstrated its usefulness with highly reduced computational burden in the test case applied to a hexapod under locked joint failure compared to traditional disturbance rejection methods, while the system maintains stability conditions. Future work will be toward improving the self-organization method through deeper analysis in resilience, focusing on the vulnerability and recoverability of systems under failure modes.

REFERENCES

Vinerbi L., Bondavalli A. & Lollini P. (2010). Emergence: A new Source of Failures in Complex Systems. IEEE Third International Conference on Dependability (133-138), July 18-25, DOI: 10.1109/DEPEND.2010.28

Prehofer C. & Bettstetter C. (2005), Self-organization in communication networks: principles and design paradigms, IEEE Communications Magazine, July 25, DOI: 10.1109/MCOM.2005.1470824

Heylighen F. (1999), The Science of Self-Organization and Adaptivity, The Encyclopedia of Life Support Systems, Vol. 5 (No. 3), 253-280

Wilson R. (1996). Introduction to Graph Theory. Edinburgh Gate, Harlow, Essex CM20 2JE, England: Longman.

Liu W., Sirisena H., Pawlikowski K. & McInnes A. (2009), Utility of algebraic connectivity metric in topology design of survivable networks, Design of Reliable Communication Networks, 2009. DRCN 2009. 7th International Workshop, 25-28 Oct. 2009, DOI: 10.1109/DRCN.2009.5340016

Butler S. K., (2008). Eigenvalues and Structures of Graphs. Doctoral dissertation. University of California, San Diego, http://orion.math.iastate.edu/butler/PDF/dissertation.pdf

Bellman R. (1957), A Markovian Decision Process, Indiana University Mathematics Journal, Vol. 6 (No. 4), 679-684

Yukalov V.I. & Sornette D. (2014), Self-organization in complex systems as decision making, Adv. Complex Syst., 17 (2014) 1450016

Gabbai J., (2005). Complexity and the Aerospace Industry: Understanding Emergence by Relating Structure to Performance using Multi-Agent Systems. Doctoral dissertation. University of Manchester, http://gabbai.com/files/J%20M%20E%20Gabbai%20EngD%20Thesis.pdf

Cully A., Clune J., Tarapore D. & Mouret J. (2015), Robots that can adapt like animals, Nature, Vol. 521 (No. 7553), 503-507, doi:10.1038/nature14422

Sorin M., Mircea N. & Viorel S. (2011), Hexapod robot: Mathematical support for modeling and control, System Theory, Control, and Computing (ICSTCC), 2011 15th International Conference, Oct 14-16

Barai R., Saha P. & Mandal A. (2013), SMART-HexBot: a Simulation, Modeling, Analysis and Research Tool for Hexapod Robot in Virtual Reality and Simulink, AIR '13 Proceedings of Conference on Advances In Robotics, doi>10.1145/2506095.2506126

Yang J. (2003), Fault-tolerant gait generation for locked joint failures, Systems, Man and Cybernetics, 2003. IEEE International Conference, Oct. 8, DOI: 10.1109/ICSMC.2003.1244216

Cuaya-Simbro G. & Munoz-Melendez A. (2008), Adaptive Locomotion for a Hexagonal Hexapod Robot Based on a Hierarchical Markov Decision Process, WSPC – Proceedings, Vol. 0 (No. 12), June 2008

Chades I., Chapron G., Cros MJ., Garcia F., Sabbadin R. (2014). MDPtoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems. Ecography 37:916-920.

Jeong W., Kim H., Kim S. & Jun B. (2013), Path Tracking Controller Design of Hexapod Robot for Omni-directional Gaits, Control Conference (ASCC), 2013 9th Asian, 23-26 June 2013, DOI: 10.1109/ASCC.2013.6606206

BIOGRAPHIES

Benjamin Lee is an Electrical Engineering graduate student at the Georgia Institute of Technology since 2013. He obtained his Bachelor's degree in Electrical Engineering from the Georgia Institute of Technology in 2013. He has participated in researches of structural health monitoring using acoustic waves, developing and testing of user interfaces for situation awareness in life support systems, and developing tracking control system for a unicycle mobile robot. His main research areas include resilient system design methodology and self-organizational control methods.

Sehwan Oh is a Ph.D. candidate in the ASDL at the Georgia Institute of Technology since 2010. He has participated in graduate researches of a turbine engine model regression analysis, Navy transformable ship design, risk analysis of the integration of unmanned aerial vehicle systems into the national airspace system, and smart and sustainable campus design and analysis. His main research areas include resilience system design methodology and control reconfiguration.

Michael Balchanos is a research faculty with the School of Aerospace Engineering, where he serves as the Resilient Systems Branch lead at the Aerospace Systems Design Laboratory. His areas of expertise include research work in dynamic systems modeling and simulation methods, as well as SoS-level integration techniques for enabling decision support in complex systems design, involving several applications such as smart energy infrastructures, electric reconfigurable naval ships and unmanned aerial vehicles. He is also leading ASDL's Automotive Systems Research Initiative with applications in electric vehicle energy-based sizing and optimization (EVs) as well as the development of SoS-level frameworks for the connected autonomous mobility ecosystem of the future. He obtained his Diploma in Physics from the Aristotle University of Thessaloniki, Greece and his M.Sc. and Ph.D. degrees in Aerospace Engineering from Georgia Tech.

George Vachtsevanos is a Professor Emeritus of Electrical and Computer Engineering at the Georgia Institute of Technology. He was awarded a B.E.E. degree from the City College of New York in 1962, a M.E.E. degree from New York University in 1963 and the Ph.D. degree in Electrical Engineering from the City University of New York in 1970. He directs the Intelligent Control Systems laboratory at Georgia Tech where faculty and students are conducting research in intelligent control, fault diagnosis and prognosis of large-scale dynamical systems, and control technologies for Unmanned Aerial Vehicles. His work is funded by government agencies and industry. He has published over 240 technical papers and is a senior member of IEEE. He was awarded the IEEE Control Systems Magazine Outstanding Paper Award for the years 2002-2003 (with L. Wills and B. Heck). He was also awarded the 2002-2003 Georgia Tech School of Electrical and Computer Engineering Distinguished Professor Award and the 2003-2004 Georgia Institute of Technology Outstanding Interdisciplinary Activities Award.