# Adaptive Gait Generation for Hexapod Robot using Genetic Algorithm

Aditya Manglik, Kunal Gupta

Department of Electrical and Electronics Engineering
Birla Institute of Technology And Science
Pilani, Rajasthan 333-031

Dr. Surekha Bhanot
Professor
Department of Electrical and Electronics Engineering
Birla Institute of Technology And Science
Pilani, Rajasthan 333-031

***Abstract*— Animals display naturally robust locomotion designs which enable them to move on rugged terrains and even adapt to injuries. Most of the robots do not possess such robustness and are rendered useless if they get structurally damaged. In this paper, it is shown that such adapting ability can be introduced in robot's locomotion, if damaged, by evolving optimal gaits through genetic algorithm (GA). A hexapod robot simulation was used to test the gaits generated by GA. Simulation results have proven that the evolved gait enables a hexapod to move effectively with one leg damaged. This technique will increase the reliability and effectiveness of autonomous robots in areas hostile to humans.**

**Keywords- gait generation, hexapod, genetic algorithm, damage recovery, adaptive gait**

## I. Introduction

Autonomous robots are involved in a wide range of activities in homes, offices, manufacturing industries etc. They are often used in areas hostile to humans such as outer space, deep oceans, nuclear power plants, sewage treatment plants, disaster struck areas etc. A robot's locomotion design refers to its ability to navigate the surroundings. Locomotion designs exist in a large variety, from wheels to belts and even limbs. Limbed robots have the ability to navigate areas which are deprived of smooth surfaces. Rough, rocky, steep and hostile terrains make their use inevitable owing to this unique locomotion design.
However, these robots tend to fail miserably if they suffer some structural damage due to a mishap. Nature has found ways to adapt even in extreme cases of limb loss. Animals like cats, dogs miraculously adapt to a new walking mechanism, almost instantaneously after damage to their limb(s). On the contrary, their mechanical imitations i.e. robots are mostly useless due to inability to adapt to damage. This flaw is design based and potentially causes huge losses to the industry annually.
For adaptation to limb loss, the robot must develop a new gait. Generating a gait that ensures stability and freedom to move in arbitrary directions is a complex optimization problem, as large amount of control and co-ordination among legs is required.
Genetic Algorithm mimics the principles of natural evolution and survival of the fittest. It involves a random population where each member of the population represents a possible solution. The population members are subsequently crossbred among best solutions which are then passed to the next generation. Mutations enable better solutions to evolve from existing solutions by introducing random changes in them, increasing the search space. The process of cross breeding continues iteratively till changes in population members become insignificant over a large number of generations.
This paper describes the use of GA to generate gait for a hexapod robot, using each member of population as a possible gait. Conditions for gait generation is structural damage to hexapod which has rendered a leg useless. Consequently, a new gait must be generated for the robot to be able to walk with remaining undamaged legs.The mathematical model used for simulation and GA calculations has been built using MATLAB. The best possible gait generated by the simulation is then applied to a real hexapod for testing.
Section II on hardware model briefly discusses the physical hexapod. Details about the mathematical model are given in section III. Section IV describes the way gait for the hexapod is defined and is subsequently generated in section V. Section VI and VII discuss the results obtained and concluding remarks along with avenues for further study.

Fig. 1. Natural Adaptive Gait Generation

**978-1-4673-8587-9/16/$31.00 2016 IEEE**

## II. Hardware Model

The hardware model has been designed using SolidWorks version 2014 as shown in figure 2. The robot has six legs attached to the body in a hexagonal fashion, each leg possessing three joints for movement. These three joints have been named as 'Pelvis, Knee and Feet'. The pelvis is attached to hip, hip attached to knee via femur (bone) and knee attached to feet via tibia (bone). Motion of pelvis motors (attached between top of leg and body), allows the robot to move the leg in forward direction. Motion of hip and knee motors provides upwards and downwards motion. This configuration results in three Degrees of Freedom(DOF), allowing the hexapod to move freely in a 3-D space. Servo motors have been used to move joints in required directions. One motor represents each joint, totaling $6\ X\ 3\ =\ 18\ motors$.

Fig. 2. Solidworks model of hardware robot

Motor control is done via an Arduino Mega board. The Arduino is interfaced with MATLAB for implementation of GA. Physical limitations imposed by high trial time for each iteration and large number of iterations made online implementation of GA unfeasible. Therefore, gait generation is initially simulated using a MATLAB model. The final evolved gaits are implemented on the hexapod.

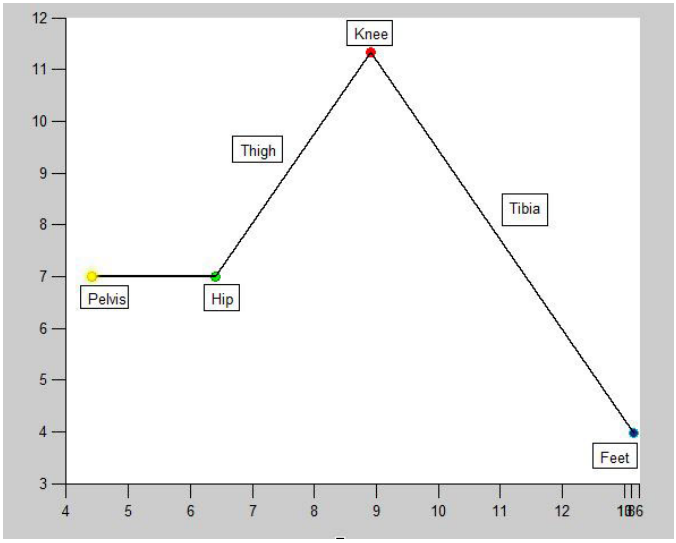


Fig. 3. Simulated Leg

## III. Mathematical model

MATLAB has been used to simulate the mathematical model of the hexapod. The mathematical model is built upon kinematic equations for the hexapod and does not consider more complex dynamics of motion. Due to this limitation, actions such as jumping cannot be simulated. Yaw, pitch and roll of the hexapod's body are also ignored. Consequently, center of the hexapod stays at constant height throughout motion. The model assumes following condition for stability of robot: *Projection on ground of center of gravity of the robot must always lie inside the polygon defined by feet of legs currently on ground. Else, the robot will become unstable.* Simulation has been done by taking into account angles and co-ordinates of all joints, along with co-ordinates of feet. The center of hexapod is equidistant from all pelvis motors. The forward direction of motion coincides with the positive x-axis of a right handed co-ordinate system. Simulated leg of the model is shown in figure 3. The parameters used in mathematical modelling for $i^{th}$leg are described as follows-

### A. Parameters

$$\delta_i = Angle\ made\ by\ each\ Pelvis\ motor\ with\ x-axis \tag{1}$$

$$\phi_i = Smaller\ angle\ made\ by\ hip\ motor\ with\ y-axis \tag{2}$$

$$\tau_i = Complement\ of\ angle\ made\ by\ hip\ motor\ with\ z-axis \tag{3}$$

$$\theta_i = Angle\ between\ tibia\ and\ thigh \tag{4}$$

$$h_i = Length\ of\ Hip \tag{5}$$

$$t_i = Length\ of\ Thigh \tag{6}$$

$$b_i = Length\ of\ Tibia \tag{7}$$

$$l_i = Distance\ between\ Pelvis\ and\ center\ of\ Robot \tag{8}$$

$\delta_i,\ l_i,\ h_i,\ t_i,\ b_i$ are constant.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}\ are\ the\ co-ordinates\ of\ center\ of\ robot. \tag{9}$$

### B. Equations

$Equations\ for\ motion\ of\ Pelvis\ (x_{pi}, y_{pi}, z_{pi})\ :$

$$x_{pi} = x_c + l_i cos(\delta_i) \tag{10}$$

$$y_{pi} = y_c + l_i sin(\delta_i) \tag{11}$$

$$z_{pi} = z_c \tag{12}$$

Motion due to Pelvis allows the robot to move entire leg forward and backward.

$$Equations\ for\ motion\ of\ Hips\ (x_{hi}, y_{hi}, z_{hi})\ :$$

$$x_{hi} = x_{pi} + h_i sin(\phi_i) \quad (13)$$
$$y_{hi} = y_{pi} + h_i cos(\phi_i) \quad (14)$$
$$z_{hi} = z_{pi} \quad (15)$$

Motion due to Hip allows the robot to move thigh upwards and downwards.

$$Equations\ for\ motion\ of\ Legs\ (x_{ki}, y_{ki}, z_{ki})\ :$$

$$x_{ki} = x_{hi} + t_i cos(\tau_i)\ sin(\phi_i) \quad (16)$$
$$y_{ki} = y_{hi} + t_i sin(\tau_i)\ cos(\phi_i) \quad (17)$$
$$z_{ki} = z_{hi} + t_i sin(\tau_i) \quad (18)$$

Motion due to Legs allows the robot to move the tibia upwards and downwards.

$$Equations\ for\ motion\ of\ feet\ (x_{fi}, y_{fi}, z_{fi})\ :$$

$$x_{fi} = x_{ki} + b_i sin(\theta_i)\ sin(\phi_i) \quad (19)$$
$$y_{fi} = y_{ki} + b_i sin(\theta_i)\ cos(\phi_i) \quad (20)$$
$$z_{fi} = z_{ki} - b_i sin(\tau_i) \quad (21)$$

Motion of feet describe the co-ordinates of foot on the ground for each leg.
In order to move the robot, the translation of feet on the ground has been taken into account.

$$\begin{bmatrix} \Delta x_c \\ \Delta y_c \\ \Delta z_c \end{bmatrix} = \alpha \sum_{1}^{6} \begin{bmatrix} x_{fi}(final) - x_{fi}(initial) \\ y_{fi}(final) - y_{fi}(initial) \\ z_{fi}(final) - z_{fi}(initial) \end{bmatrix} \quad (22)$$

These equations determine the displacement of center co-ordinates from their initial values. The displacement is used to calculate final values of center of base.
$\alpha \in (0, 0.5]$ is a normalizing constant. The model thus generated is shown in figure 4.

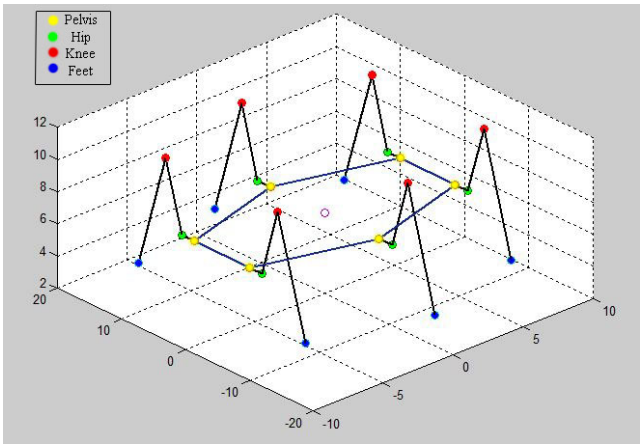


Fig. 4. Mathematical model simulation

## IV. Description of Gait

Gait (walking mechanism) is defined as a sequence of consequent steps where every following step is a derivative of the state of legs from the previous step. Development of gaits for a hexapod requires control and co-ordination of six simultaneously moving legs. Every gait must have an initial step. After the initial step is defined, successive steps are derived from their respective preceding steps. They are looped in order to minimize the complexity of computation tasks and provide continuous transfer from last step to first step without compromising stability. State of each leg is defined in terms of 3 angles-

$$State\ of\ leg_i = \begin{bmatrix} \phi_i \\ \tau_i \\ \theta_i \end{bmatrix} \quad (23)$$

Each step is defined as a row vector of six states, one state for each of hexapod's six legs.

$$Step = \begin{bmatrix} State\ of\ leg_1 & \dots & State\ of\ leg_6 \end{bmatrix} \quad (24)$$

State values have been quantized in order to simplify the process of gait generation and minimize the possibility of generation of unstable states. Every leg can have five possible states. These include three support state values and two transition state values. Support states include those in which the leg touches the ground and supports weight of the robot. Transfer states include those in which the leg is lifted and is being moved to a new position.
State values have been defined as:

- DR - (Down Rear) leg is on the ground in backward position, support state
- DC - (Down Center) leg is on the ground in central position, support state
- DF - (Down Forward) leg is on the ground in forward position, support state
- UF - (Up Forward) leg is in the air in forward position, transfer state
- UR - (Up Rear) leg is in the air in rearward position, transfer state

Increment in states follows the cycle as shown in figure 5.

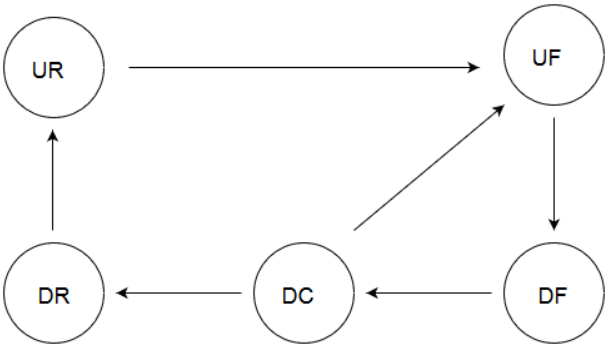


Fig. 5. Order of Transfer of States

Each gait is initialized from step 1 having all state values set to DC. Step 2 involves transition from DC to UF. Step 3 onwards, gait generation follows the sequence:

$$UF \rightarrow DF \rightarrow DC \rightarrow DR \rightarrow UR \rightarrow UF$$

At every state increment, stability of transition with respect to prior state is checked. This process is followed for all steps of gait generation.

$$\begin{aligned} Step\ 1 &= \begin{bmatrix} State\ of\ leg_1 & \dots & State\ of\ leg_6 \end{bmatrix} \\ Step\ 2 &= \begin{bmatrix} State\ of\ leg_1 + 1 & \dots & State\ of\ leg_6 + 1 \end{bmatrix} \\ &\vdots \\ Step\ N &= \begin{bmatrix} State\ of\ leg_1 + N & \dots & State\ of\ leg_6 + N \end{bmatrix} \end{aligned}$$

Gait is defined as a column vector of N steps where each step is derived from the previous step.

$$Gait = \begin{bmatrix} Step\ 1 \\ \vdots \\ Step\ N \end{bmatrix} \tag{25}$$

Thus, a gait represents a $[3N$ x $6]$ matrix, N being the number of steps. GA is used to determine these 18N elements to optimize the walking mechanism of the spider.A gait having 2 steps is given below for moving legs 1 and 6 from DC to UF. Each value in the matrix represents a specific angle for servo motor as per Equations (23) & (24). Consequently, one step has 18 values for moving the 18 motors of hexapod.

$$gait = \begin{bmatrix} 00 & 00 & 00 & 00 & 00 & 00 \\ 60 & 60 & 60 & 60 & 60 & 60 \\ 30 & 30 & 30 & 30 & 30 & 30 \\ 00 & 00 & 30 & 00 & 00 & 30 \\ 60 & 60 & 60 & 60 & 60 & 60 \\ 30 & 30 & 45 & 30 & 30 & 45 \end{bmatrix}$$

Experimentation with the hardware robot has been used to determine the optimal value of N, which has been found to be twelve.

## V. Application of Genetic Algorithm for Gait Generation

Basic description of GA is given in figure 6. Initially a random population of individuals, each representing a gait, is created. Each gait's performance is simulated in MATLAB and its fitness is evaluated on the following criteria:

1) Forward displacement
2) Sideways displacement

The above two factors are normalized with respect to number of steps in gait.

- Greater forward displacement, higher fitness value.
- Greater sideways displacement, lesser fitness value.

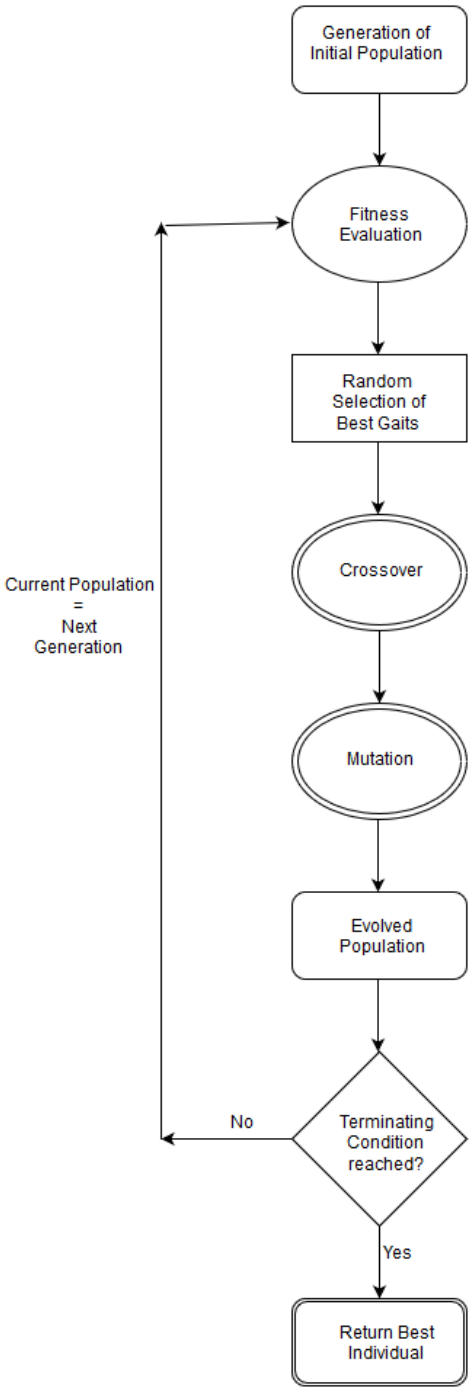


Fig. 6. Description of genetic algorithm for gait generation

### A. Crossover

To get a stable gait as an offspring of two parent gaits, the method of Vertical One-Sided Transfer(VOT) is employed. In this method, the state values of randomly chosen legs are swapped with respective state values from legs of second gait. Swapping of state values is done throughout the gait, resulting in breeding of two parents to produce an offspring.

Since the offspring gait is likely to be unstable, a stability check is required. The offspring is passed to the next population only if it has a better fitness value than each of its parent gaits, otherwise the parent gait with greater fitness values is passed to the next generation. This is done to increase the convergence of GA and reduce the number of iterations.
Crossover algorithm in pseudo-code form is given below. The $stable\_check$ function checks each step of the gait with regard to the stability condition specified in Section III. The function returns a value of zero for an unstable gait and a value of one for a stable gait.

**Title**: Crossover Algorithm
**Input**:Parent gaits, ($gait_1$ and $gait_2$)
**Output**: Crossed over $gait_3$

initialization;
stability=0;
**while** *(*$stability$ *= 0)* **do**
  $offspring$ = crossover of $gait_1$ , $gait_2$ ;
  % Using VOT
  $stability$ = stable_check($offspring$);
**end**
**if** *(* *fitness(*$offspring$*) > fitness(*$gait_1$*) )* *and* *(fitness(*$offspring$*) > fitness(*$gait_2$*) )* **then**
  $gait_3$ = $offspring$ ;
**else**
  **if** *(* *fitness(*$gait_1$*) > fitness(*$gait_2$*) )* **then**
    $gait_3$ = $gait_1$ ;
  **else**
    $gait_3$ = $gait_2$ ;
  **end**
**end**
return $gait_3$ ;

### B. Mutation

In order to keep the algorithm from getting stuck in local optima and to expand the search space, mutations are required. To implement mutation in a gait, step slicing method is used. Gait is sliced at a random step and part of gait before the sliced step is retained. Other part is obtained by randomly incrementing state values of each leg after the sliced step, till the required number of steps in gait are reached. This mutated gait is passed on to the next population only if it is stable and possesses a better fitness value as compared to the initial gait. Pseudo-code description of the mutation algorithm is given below.

### C. Gait generation for damaged robot

Structural damage to robot is modelled in simulation as a leg which is unresponsive to any command. Hence, state of the damaged leg does not change at any step in gait. However, the damaged leg may be used for supporting the body. Normal motion is affected because of the damaged leg, as the robot tends to deviate sideways. Since one of the legs is not contributing to the forward motion, forward displacement is also reduced.

**Title**: Mutation Algorithm
**Input**:Gait to be mutated, $gait$
**Output**: Mutated gait, $mutated\_gait$

initialization;
stability=0;
**while** *(*$stability$ *= 0)* **do**
  $mutation$ = mutation of $gait$;
  %Using step slicing method
  $stability$ = stable_check($mutation$);
**end**
**if** *( fitness(*$mutation$*) > fitness(*$gait$*) )* **then**
  mutated_gait = $mutation$ ;
**else**
  mutated_gait = $gait$ ;
**end**
return $mutated\_gait$ ;

GA is again applied to the simulation model after introducing structural damage. Results thus obtained have been discussed in the next section.

## VI. Results

The objective of this research has been to demonstrate that GA can generate stable gaits even when the robot is structurally damaged. To obtain the best possible gaits, a number of combinations for population size and number of generations have been tested. Results corresponding to population size of 50 and number of generations as 40 have shown promising results.

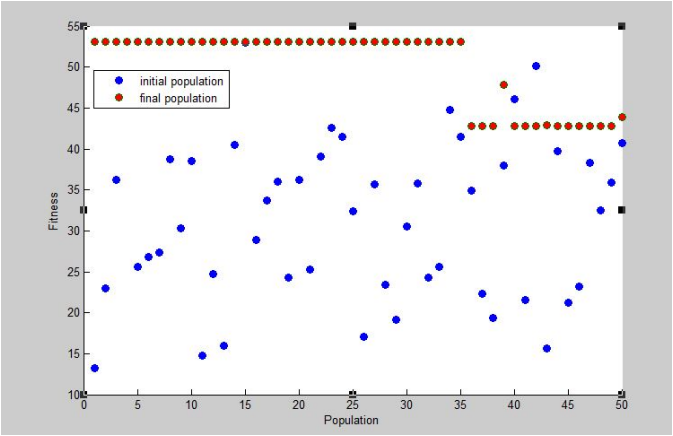


Fig. 7. Initial vs final evolved population

Figure 7 compares fitness values of initial and final evolved population. Blue particles represent randomly generated initial population, while the red particles show evolved population after application of GA. Evolved populations consistently displayed very high fitness values. This shows that GA is successful in maximizing the fitness of gait.

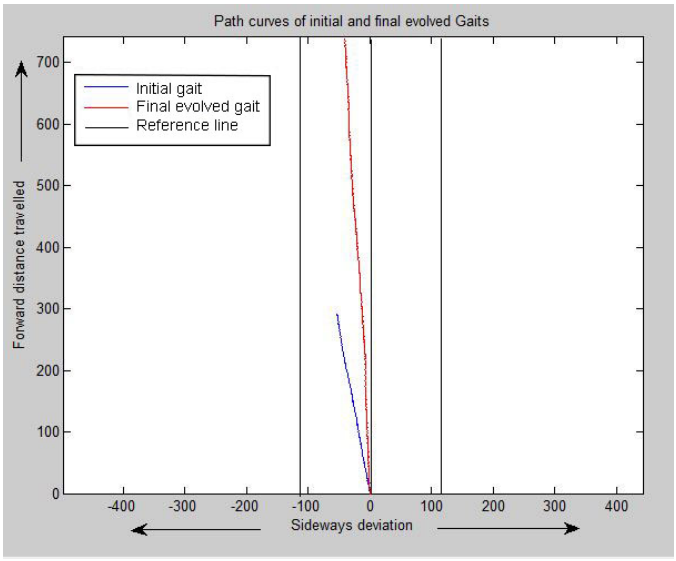


Fig. 8. Comparison of initial and final evolved gaits

Figure 8 compares the trajectories of initial and final(evolved) gaits for undamaged robot. The final evolved gait traverses 2.64 times the first generation gait's distance for the same number steps. Sideways deviation for the same distance travelled is reduced by 62.5% for the evolved gait.

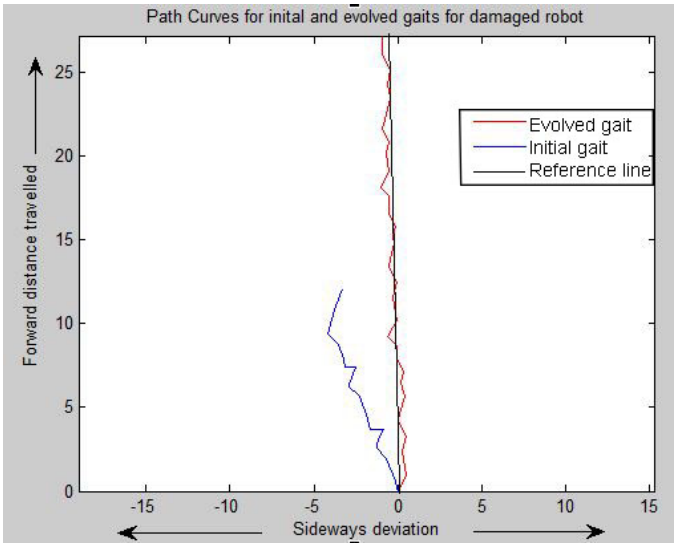


Fig. 9. Path Curve for initial and evolved gaits of damaged robot

Figure 9 compares trajectories for initial and evolved gaits of the damaged robot. Initially when damage is inflicted to forward leg, the robot deviates sideways by upto 5 units for 12 units of forward motion, resulting in poor performance. After evolution, the robot learns to use its undamaged legs to counter the sideways deviation caused by damaged leg.

Figure 10 compares the trajectories of final evolved gaits for damaged and undamaged robot. The path covered by evolved gait for damaged robot differs sideways by 17% and

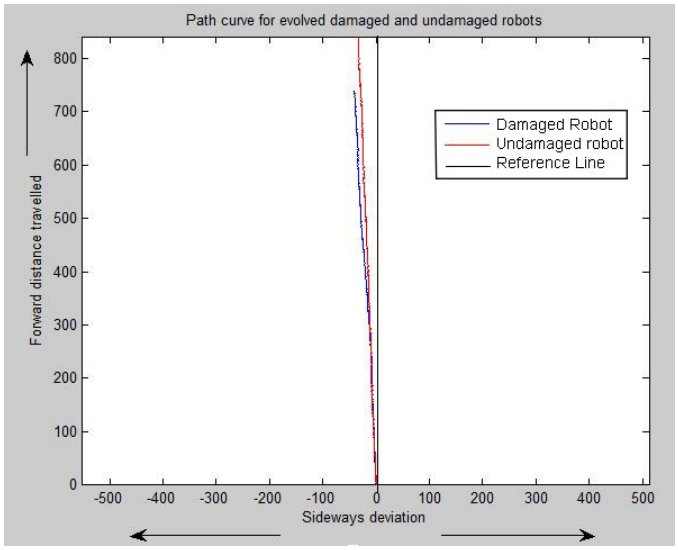


Fig. 10. Comparison between damaged and undamaged gaits(evolved)

forwards by 12% from that of the undamaged robot, confirming that the robot has indeed adapted to motion with damaged leg.

## VII. Conclusions

The algorithm is successful in evolving gaits to adapt to physical damage and performs close to the undamaged robot. Best gaits evolved from the MATLAB simulation have been applied to real hexapod and some inefficiencies were observed. This is due to lack of support for friction, fatigue due to constant usage and impulsive motion, in the MATLAB model. Future research may be done in improving the model and simulation conditions to minimize these inefficiencies.

## References


[1] C. Darwin. *Origin of Species*, London, UK: John Murray, 1859.

[2] F. Seljanko, *Hexapod walking robot Gait Generation using Genetic Gravitational Hybrid algorithm*, 15th International Conference on Advanced Robotocs, 2011.

[3] Spencer, G,1994. *Automatic Generation of Programs for Crawling and Walking*, Advances in Genetic Programming. (pp. 335-353) K. Kinnear, Jr. (ed.), Cambridge, MA: MIT.

[4] Gary B. Parker, William T. Tarimo, and Michael Cantor. *Quadruped Gait Learning Using Cyclic Genetic Algorithm*, in Proceedings of 2011 IEEE Congress on Evolutionary Computation (CEC 2011).

[5] J. Currey,M., Beckerleg, J. Collins. *Software Evolution Of A Hexapod Robot Walking Gait*, 2008. 15th International Conference on Mechatronics and Machine Vision in Practice, 2008. M2VIP, pp. 305 - 310.

[6] V. Dturr, J. Schmitz, and H. Cruse. *Behaviour-based modelling of hexapod locomotion: Linking biology and technical application*, Arthropod Structure & Development, vol. 33, pp. 237-250, 2004.

[7] G. Parker, D. Braun, and I. Cyliax. *Evolving Hexapod Gaits Using Cyclic Genetic Algorithms*, New London, CT: Indiana University.

[8] Mahdavi and P. Bentley. *An evolutionary approach to damage recovery of robot motion with muscles*, Advances in Articial Life, pages 248255, 2003.

[9] J.C. Bongard, H. Lipson. *Automated damage diagnosis and recovery for remote robotics*