Optimal Gait Analysis of Snake Robot Dynamics [*]

Vipul Mehta
Department of Mechanical Engineering
The Pennsylvania State University
University Park, PA - 16802
vvm104@psu.edu

ABSTRACT

Though there have been a lot of research in the area of snake-robot kinematics and dynamics, a little attention has been given to find out an optimal gait for the robot. This optimal gait until now is being calculated using a graphical method. An attempt, here, is made to get these optimum gait parameters using evolutionary algorithms.

We intend to optimize the input power consumed by the robot for a given propulsive speed. A popular multi-objective evolutionary algorithm developed by Deb et al., NSGA-II is used in this work and the results are presented.

Results from an approximation of objective function through polynomials and from the actual simulation are presented. Two different frictional models are considered and their results are given. The results are in good agreement with the literature. A parametric study is also included to find minimum population size and number of generations. The performance metrics are used to justify the parametrization.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*Graphical Analysis*; I.2.9 [**Artificial Intelligence**]: Robotics—*Kinematics and Dynamics*

General Terms

Performance

Keywords

Snake-robot, NSGA-II, Evolutionary Algorithms and robotics, Pareto front

1. INTRODUCTION

Lot of research has been done in the area of snake-robots since Hirose [6]. Primary use of snake-robots is in surveillance in rough terrains (e.g. deserts, mountains, pipes) where wheeled, legged robots tends to be inefficient [4]. The ability to move around fast with least power consumption is an important issue for any autonomous robot. But a little attention has been given to find out the optimal gait of the robot under different operating conditions. This involves finding the gait parameters that yield minimum power consumption for a given forward speed. The power input and the forward speed conflicts with each other. The efforts reported so far [4, 10, 8] are based on a graphical technique. In this method the search area is meshed into a fine grid and the objective functions are calculated at these points. The Pareto front and therefore the optimal gait parameters are decided comparing the objective function at each of the values. The accuracy of the method depends on the resolution of the grid considered. As the resolution is made finer, the search becomes more exhaustive but it takes longer computational time for the optimization process.

Finding the Pareto front is a difficult problem for classical optimization [1]. The most popular method involves weighted sum of the objective functions. The main drawback of this technique is the scaling issue. To overcome this difficulty one needs to know the bounds on the objective function, which may not be always possible. Moreover the classical optimization methods demand the gradient of the objective function. This is also hard to obtain for an implicit, nonlinear objective function.

The problem, we are going to discuss, possesses both the above described problems. There may exist a few orders of magnitude difference between two objective functions and because of the non-linear, implicit governing equations of motion the gradient information is not available. We, therefore, seek an optimization method that can give perhaps more accurate results or can provide a solution quicker for the same accuracy of the graphical method. It can also serve as a confirmation of the results obtained by the graphical method.

There has been a huge development recently in the area of evolutionary optimization where heuristic approaches are used to find an optimum solution. The biggest advantage of these methods is that they do not require the derivative of the objective function. An evolution algorithm for multi-objective problems, Non-dominated Sorting Genetic

[*]This report has been prepared for the course project of *System Optimization using Evolutionary Algorithms*, CE 563

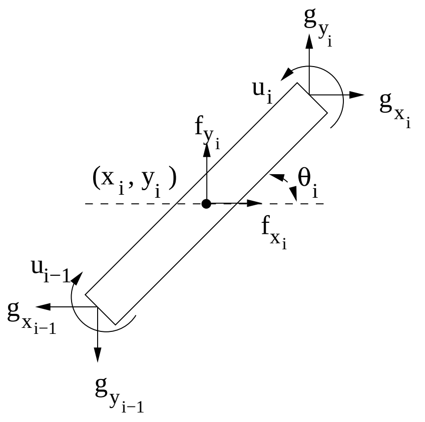

Figure 1: Free body diagram of i^{th} link

Algorithm-II (NSGA-II), has been shown to work very good with complex problems [3]. The problem of gait analysis is analyzed using NSGA-II in this project. The remaining report discusses the formulation of the objective function, polynomial approximation of the objective function, parametrization of NSGA-II and the results from the analysis of the snake-robot dynamics for two case-studies.

2. FORMULATION OF OBJECTIVE FUNCTION

The detailed dynamics of the snake-robot is given in [10]. If a robot consists of n rigid links connected together with $(n-1)$ joints, defining the coordinates of the center of mass and global orientation of each link is given by,

$$\begin{aligned} x &= [x_i]_{n\times 1} && (1)\\ y &= [y_i]_{n\times 1} && (2)\\ \theta &= [\theta_i]_{n\times 1} && (3) \end{aligned}$$

This representation is shown in Fig. 1. Let the mass, length and the mass moment of inertia of each link be m, $2 \cdot l$, and J $(= m \cdot l^2/3)$ respectively. Newton's laws are applied in x-direction as well as in y-direction and the equations are simplified. Defining the position of the center of mass of the robot as $w = 1/n \cdot [\sum x_i \quad \sum y_i]'$, the resulting governing differential equations of motion are given by

$$\begin{bmatrix} \mathcal{J} & 0 \\ 0 & m\cdot \mathbf{I}_2 \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{w} \end{bmatrix} + \begin{bmatrix} \mathcal{C}\cdot\dot{\theta}^2 \\ 0 \end{bmatrix} - \begin{bmatrix} \mathcal{L}' \\ E' \end{bmatrix} \begin{bmatrix} f_x \\ f_y \end{bmatrix} + \tau = \begin{bmatrix} D' \\ 0 \end{bmatrix} u \quad (4)$$

where,

$$\begin{aligned} \tau &= c_n (J/m)\dot{\theta} && (5)\\ \mathcal{J} &= J\cdot \mathbf{I}_n + S_\theta \cdot H \cdot S_\theta + C_\theta \cdot H \cdot C_\theta && (6)\\ H &= m\cdot l^2 \cdot \left[A' \cdot (D\cdot D')^{-1} \cdot A\right] && (7)\\ M &= n\cdot m \text{ (the total mass of the robot)} && (8)\\ \mathcal{C} &= S_\theta \cdot H \cdot C_\theta - C_\theta \cdot H \cdot S_\theta && (9)\\ \mathcal{L} &= \left[S_\theta \cdot N' \cdot -C_\theta \cdot N'\right]' && (10)\\ N &= l\cdot \left[D' \cdot (D\cdot D')^{-1}\cdot A\right] && (11)\\ E &= \begin{bmatrix} e & 0 \\ 0 & e \end{bmatrix} && (12)\\ e &= [1 \cdots 1]' && (13) \end{aligned}$$

and $\mathbf{I}_n$ is the identity matrix of dimension n and u is the torque input vector applied at the joints. Moreover

$$\begin{aligned} S_\theta &= \operatorname{diag}(\sin\theta_1, \cdots, \sin\theta_n) && (14)\\ C_\theta &= \operatorname{diag}(\cos\theta_1, \cdots, \cos\theta_n) && (15) \end{aligned}$$

$$D = \begin{bmatrix} 1 & -1 & & 0 \\ & \ddots & \ddots & \\ 0 & & 1 & -1 \end{bmatrix}_{(n-1)\times n} \quad (16)$$

$$A = \begin{bmatrix} 1 & 1 & & 0 \\ & \ddots & \ddots & \\ 0 & & 1 & 1 \end{bmatrix}_{(n-1)\times n} \quad (17)$$

Also $[f_x \quad f_y]'$ are the frictional forces acting on i^{th} link. There are various friction models considered in the literature (e.g. see [8]). In this report, we consider only the viscous friction model, where the frictional forces are proportional to the velocity of the center of mass of the link. These forces are then represented as

$$\begin{bmatrix} f_x \\ f_y \end{bmatrix} = -\Omega_\theta \cdot D_f \cdot \Omega_\theta' \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} \quad (18)$$

where

$$\begin{aligned} D_f &= \begin{bmatrix} c_t \cdot \mathbf{I}_n & 0 \\ 0 & c_n \cdot \mathbf{I}_n \end{bmatrix} \\ \Omega_\theta &= \begin{bmatrix} C_\theta & -S_\theta \\ S_\theta & C_\theta \end{bmatrix} && (19) \end{aligned}$$

where c_t and c_n are the tangential and normal coefficients of friction.

The governing equations of motion given in Eqn. 4 is nonlinear, implicit differential equations, which may not be solvable directly for a given torque input. These equations are modeled in Simulink and MATLAB's in-built ODE solver (`ode45`) is used to numerically calculate the coordinates of the center of mass and the orientations of the links for a given torque. Previous research has shown [6] that the a gait similar to natural snake can be obtained if the joint angles is given by

$$\phi_i = \alpha \cdot \sin(\omega t + (i-1)\cdot\beta) \quad (20)$$

where ϕ_i is the joint angle for joint i:

$$\phi_i = \theta_{i+1} - \theta_i, \quad (21)$$

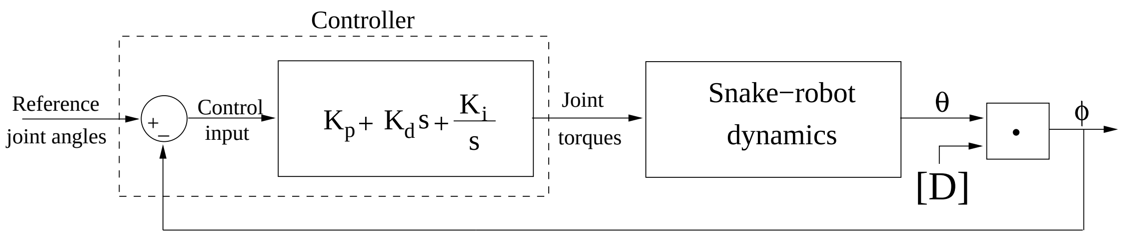

Figure 2: Block diagram showing implementation of the closed-loop controller to generate joint tracking in the simulations of snake locomotion

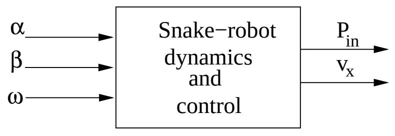

Figure 3: Structure of objective functions - inputs (α, β, ω) and outputs $(P_{\min}, v_x)$

α is the amplitude of the motion, ω is the frequency and β is the phase difference between two consecutive joint angles. The mathematical relation between the joint angles and the input torque is very difficult to find. A numerical technique is, therefore, implemented as shown in Fig. 2 using one of the simplest controller (PID - Proportional Integral Derivative controller). The control parameters are chosen so that the actual trajectory follows the reference trajectory very closely.

The objective functions can now easily be determined by

$$P_{\text{in}} = \text{Inputpower} = u \cdot \dot{\phi} \tag{22}$$

$$v_x = \text{Forward speed of the robot} = w(1) \tag{23}$$

The optimization problem is schematically shown in Fig. 3. It should be noted down here that the controller takes several cycles (4, in this case [8]) to get to the steady state value. Therefore the simulation is run for several (simulation) seconds. At every time-step of the simulation numerical integration is carried out, which makes the overall calculation of optimization quantities time-costly.

In the present analysis two separate case studies are considered for two different friction models. For a ‘sliding’ contact between the robot and the ground, the frictional torque (τ) in Eqn. 4 is non-zero and is given by the Eqn. 5. While if the contact between the robot and the ground is ‘rolling’, the frictional torque is negligible and τ in Eqn. 4 is taken to be zero. It has been shown that for these cases the values of the optimum gait parameters are different [8].

3. OPTIMIZATION PROBLEM

In this section the optimization problem is formally defined. A robot is considered to be efficient when it consumes minimum power at the desired speed. The only input variables to the robot are α, ω and β. There must exist a combination of these parameters, that can fulfill this requirement for a given speed. Mathematically

$$\min P_{\text{in}}$$

Subject to:

$$\begin{aligned} v_x &= v_{x_{\text{desired}}} \\ \alpha_{\min} &\le \alpha \le \alpha_{\max} \\ \beta_{\min} &\le \beta \le \beta_{\max} \\ \omega_{\min} &\le \omega \le \omega_{\max} \end{aligned} \tag{24}$$

where $v_{x_{\text{desired}}}$ is the desired speed of the robot, $\alpha_{\min}$, $\beta_{\min}$ and $\omega_{\min}$ are the lower bounds on the input variables (= 3 deg, 6 deg and 0.5 rad/s respectively) and $\alpha_{\max}$, $\beta_{\max}$ and $\omega_{\max}$ (= 90 deg, = 180 deg. and 5 rad/s respectively) are the upper bounds on the input variables.

It is evident that if the input power is fixed then it is needed to maximize the forward speed. This is the dual problem of the above stated problem and it has the same optimum solution as above [1]. The overall problem can be equivalently represented as a min-max problem with two objective functions,

$$\begin{aligned} &\min P_{\text{in}} \\ &\max v_x \end{aligned} \tag{25}$$

Subject to:

$$\begin{aligned} \alpha_{\min} &\le \alpha \le \alpha_{\max} \\ \beta_{\min} &\le \beta \le \beta_{\max} \\ \omega_{\min} &\le \omega \le \omega_{\max} \end{aligned}$$

Many available optimization algorithms are developed for min-min multi-objective problem (e.g. [3]). So the above problem can also be written as

$$\begin{aligned} &\min P_{\text{in}} \\ &\min(v_{\max} - v_x) \end{aligned} \tag{26}$$

Subject to:

$$\begin{aligned} \alpha_{\min} &\le \alpha \le \alpha_{\max} \\ \beta_{\min} &\le \beta \le \beta_{\max} \\ \omega_{\min} &\le \omega \le \omega_{\max} \end{aligned}$$

where $v_{\max}$ is the maximum speed possible speed of the robot with the given constraints on the gait parameters. One need not have to know this number exactly; it only has to be large enough to keep the objective function positive definite for all values of v_x. To summarize Eqns. 26 are used hereafter to determine the desired Pareto front and hence the optimal gait variables.

4. SOLUTION STRATEGY

This section describes the various techniques used to find the solution of the multi-objective problem described above. It first gives a general outline of a multi-objective evolutionary algorithm, NSGA-II. To simplify and speed up the analysis, an approximation of the objective function is suggested. Finally the computer implementation of these strategies are discussed.

4.1 NSGA-II

Non-dominated Sorting Genetic Algorithm is a popular evolutionary multi-objective algorithms (EMO's). It is preferred because it has been shown that NSGA-II performs better than the other EMO's [3, 9]. In this algorithm, the entire population is sorted according to their non-dominance over the other solutions. Here a solution is dominated means there exist a solution, which yields all the objective functions better. All the solution on the Pareto front are non-dominated. Based on the number of dominated solutions, a rank is assigned to a particular member of the population. A front comprises of the members having the same rank. The front consisting members of rank 1 is the current best estimate of the Pareto front. To preserve diversity among the population of same generation, the crowding distance approach is used. The crowding distance is the largest cuboid that encloses the current member but using the members on the same front. The selection operator used is tournament selection where both children and parents in the same generation compete each other. The members having a better rank gets selected automatically. If any two members has the same rank, one having greater crowding distance is selected. This process is iterated till a sufficient number of generations. Note that there is no convergence criteria except the number of generations. Therefore it is important to set an appropriate population size and maximum number of generations in this algorithm.

4.2 Polynomial Approximation

As mentioned earlier the objective function calculation involves numerical solution of ODE. It takes approximately 10-15 sec. to complete the simulation for one set of gait variables. It is therefore very time-costly to experiment with the algorithm running the simulation every time. An approximate function can be used instead to speed up the calculations. In this approximation, the entire domain (search space) is divided into a set of discrete points. The objective function value is calculated at these locations using the actual simulation and its value is stored. To find the objective function for the design variables not on these locations, polynomial interpolation is used. For a sufficiently high degree of polynomial, the approximation is close to the actual simulation. As the degree of the polynomial increases the computational time also increases. In the current case, cubic interpolation is found to be a good approximation.

4.3 Implementation

MATLAB is used as the platform for implementation. The objective function is evaluated using the procedures described in the earlier sections. The frictional coefficients, c_n and c_t, are 10 and 0.1 respectively. The mass (m) and half-length (l) are taken to be unity. The polynomial approximation is implemented using MATLAB's in-built function, `interp3` [7]. For the meshing, α is varied from 3 deg. to 90 deg. in the steps of 3 deg. , β is varied from 6 deg. to 180 deg. in the steps of 6 deg. and ω is varied from 0.5 rad/s to 5 rad/s in the steps of 0.5 rad/s. The objective function value other than these points are determined by cubic interpolation using `interp3`.

NSGA-II is freely available on Deb's website [2]. This program is written in C. MATLAB's in-built C-compiler `mex` is used to interface NSGA-II with the objective function. This requires two additional modules to the original code. The listing of these modules are given in the Appendix.

5. RESULTS

In this section some results obtained by NSGA-II are presented. Various algorithm parameters are taken from the literature. The results for two different friction models, for sliding as well for rolling, are presented. To reduce the computational load parametrization is carried out. Results for all cases are reported.

5.1 Parameter Settings

The NSGA-II algorithm requires various parameters in the program. The recommended values [3] are used in the present work. Crossover probability (p_c) is taken to be 0.9 and mutation probability (p_m) is taken to be 0.33 ($= 1/n_d$, where n_d is number of design variables, 3). Suggested distribution index for crossover and mutation operators are $\eta_c = 20$ and $\eta_m = 20$, respectively. For polynomial approximation we use the suggested population size (N) as 100. The number of generations (n_{gen}) are kept at 250 generations for polynomial approximation. For the actual simulation these values are found using parametrization as described later. $v_{\max}$ is taken to be 7.5 for all cases, as it is found, by trial-and-error and by [8, 10], that the maximum velocity attained by the given physical properties lies somewhere between 7 to 7.5 m/s.

5.2 Results from Polynomial Approximation

The above parameters are set in the NSGA-II algorithm and Pareto front is found using polynomial approximation of the objective function. The Pareto front and the design variables along the front are shown in Figs. 4, 5, 6 and 7. These results involves the two friction cases viz. sliding contact (where frictional torque is non-zero) and rolling contact (where frictional torque is zero). The results are compared with the literature ([10, 8]).

The results indicate that Pareto fronts obtained using NSGA-II is quite close to the ones from the literature. This confirms the potential of NSGA-II in the application of such problems and the accuracy of the polynomial approximation. The optimum values of α and β are different in two friction cases. This is also in accordance with the previously observed results [8]. The variation between v_x and ω is linear and a best-fit line can be found. The slopes of the best-fit lines for sliding friction case and for rolling friction case are 0.8505 and 0.8551 respectively. Results from previous study [8] suggest the slopes to be 0.8327 and 0.8841 respectively, which shows a good agreement.

Note that these results are due to a single random seed. But

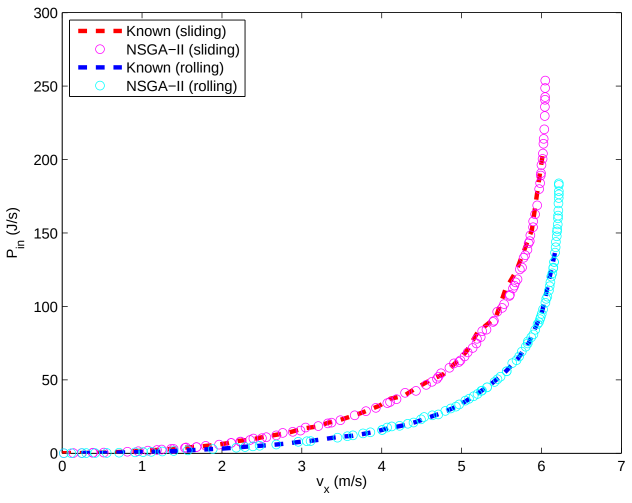

Figure 4: Estimation of Pareto front using NSGA-II and polynomial approximation. The known fronts are taken from the literature and the fronts from NSGA-II are compared with them. It shows a good agreement for both the friction cases.

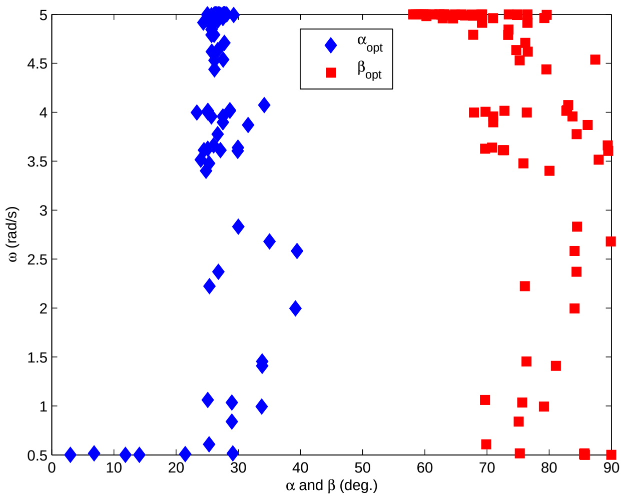

Figure 6: Design variables α and β along Pareto front using NSGA-II and polynomial approximation in the case of rolling friction. This shows that the optimum value of α remains constant around 25 deg. and the optimum value of β remains constant around 70 deg. for ω varying from 0.5-5 rad/s.

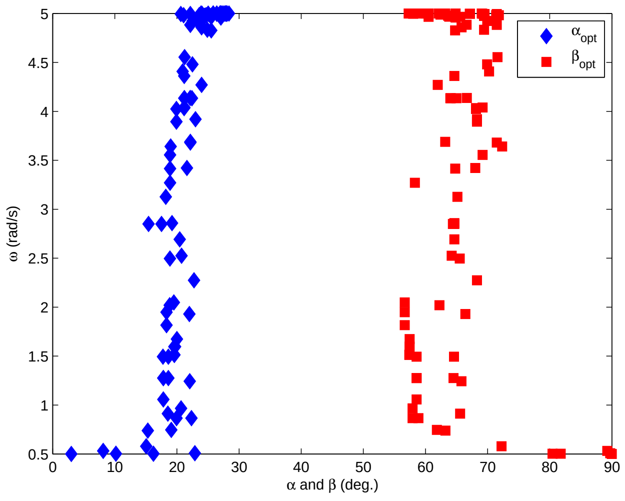

Figure 5: Design variables α and β along Pareto front using NSGA-II and polynomial approximation in the case of sliding friction. This shows that the optimum value of α increases slightly from 15 - 25 deg. and the optimum value of β remains constant around 60 deg. for ω varying from 0.5-5 rad/s.

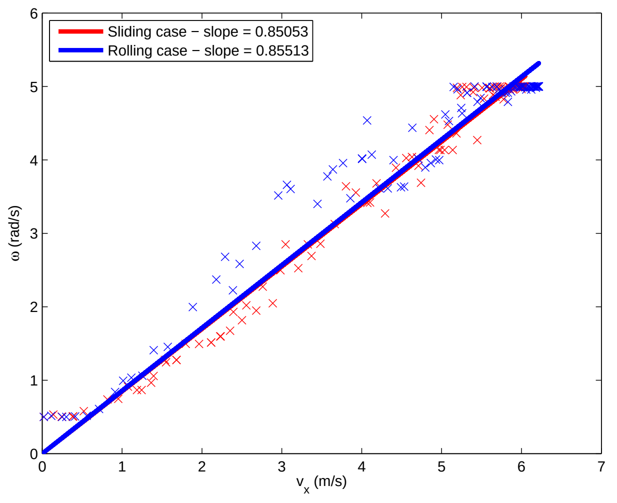

Figure 7: Design variable ω along Pareto front estimated using NSGA-II and polynomial approximation for both the frictional cases. The slope of the best-fit line is almost the same in two cases.

it is verified via numerical experiments that the results do not vary much for different random seeds.

5.3 Parametrization of NSGA-II

The objective function calculation through in-situ actual simulation takes longer computational time. The efforts described in this section are to reduce the population size and the number of generations. The experiments here are similar to that of [9].

In this process, the initial population size and maximum number of generations are kept to a small value ($N_o = 12$ and $n_{\text{gen},o} = 20$ in the present case). The algorithm is executed and the best possible Pareto front is determined. This front is compared with the known Pareto front ([8, 10]). Similar to a method described for polynomial approximation, the intermediate values are calculated using cubic interpolation. So if P_{cur} and v_{cur} are the vectors comprising the current Pareto frontier and P_{known}, $P_{v_{\text{cur}}}$ denotes the power over the known Pareto front and the cubic interpolated power for v_{cur} (using P_{known}) then the minimum square error (MSE) is defined as

$$\text{MSE} = \sqrt{\frac{1}{N} \cdot \sum^{N} \left(P_{v_{\text{cur}}} - P_{\text{cur}}\right)^2} \tag{27}$$

After every execution of NSGA-II, MSE is calculated. If the change in MSE in two consecutive iterations is greater than 5%, the population size (N) and the number of generations (n_{gen}) are incremented by N_o and $n_{\text{gen},o}$ respectively. Otherwise the process is terminated. According to Goh and Tan [5], MSE is a performance metric for the proximity indication. There are two more performance metrics namely diversity indicator (DI) and distribution indicator (S). These are adopted for the current problem as:

$$DI = \sqrt{\left[\frac{\min(P_{\text{cur}}^{\max}, P_{v_{\text{cur}}}^{\max}) - \max(P_{\text{cur}}^{\min}, P_{v_{\text{cur}}}^{\min})}{(P_{v_{\text{cur}}}^{\max} - P_{v_{\text{cur}}}^{\min})}\right]^2} \tag{28}$$

$$S = \frac{1}{N}\sqrt{\frac{1}{N} \cdot \sum^{N} \left(\text{abs}(P_{v_{\text{cur}}} - P_{\text{cur}}) - \bar{d}\right)^2} \tag{29}$$

where $\bar{d} = \sqrt{\frac{1}{N} \cdot \sum^{N} \text{abs}\left(P_{v_{\text{cur}}} - P_{\text{cur}}\right)}$ and superscripts max and min represents the maximum and the minimum value of the objective function. The metrics MSE and S should be as small as possible and DI should be close to unity.

The above procedure is run using polynomial approximation of the objective function. The graph of % change in MSE with number of parametrization cycles is shown in Fig. 8. The results indicate that the change in MSE is less than 5 % after 4 iterations. This suggests the population size and number of generations to be 48 and 80 respectively. The population size and the number of generations are significantly lower (52 % and 68% respectively) than the recommended by Deb et al [3]. Fig. 9 shows that not only all the performance metrics gets to a steady-state value but to an acceptable value. This observation justifies the use of parametrization.

5.4 Results from Actual Simulation

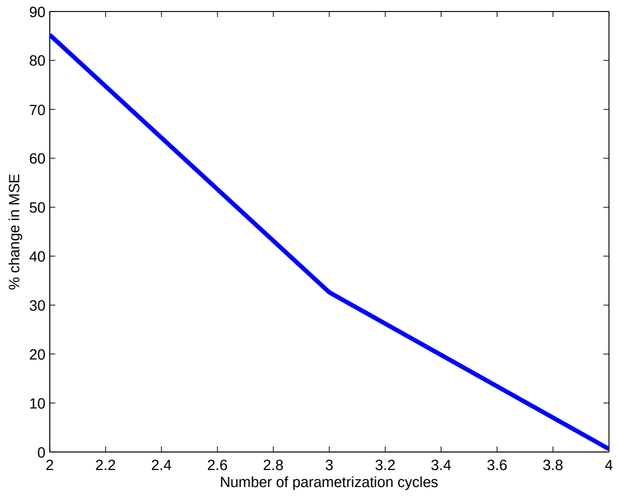

Figure 8: The variation of % change in MSE over the parametrization cycles. It has a decreasing trend. This is expected with higher population and more number of generations.

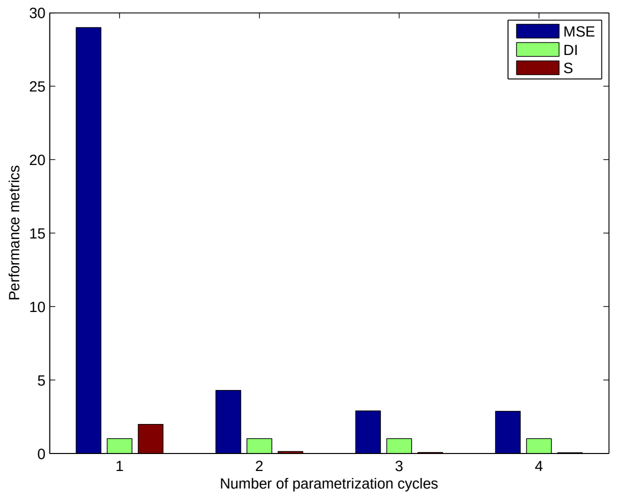

Figure 9: The various performance metrics over the parametrization cycles. The figure indicates that all the performance metrics obtains a steady-state value after 4 cycles.

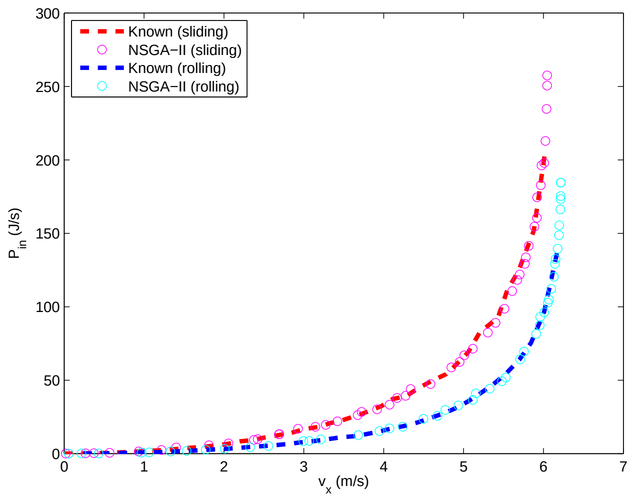

Figure 10: Estimation of Pareto front using NSGA-II and actual simulation. The known fronts are taken from the literature and the fronts from NSGA-II are compared with them. It shows a good agreement for both the friction cases.

Table 1: Performance Metrics for Different Frictional Cases and for Different Methods of evaluating Objective Function

		MSE	DI	S
Sliding friction	PA	2.8592	1.0000	0.0229
	AS	2.1895	1.0000	0.0191
Rolling friction	PA	1.2107	0.9999	0.0104
	AS	1.0295	0.9999	0.0090

Using reduced population size and the number of generations (48 and 80 respectively), the results for a single random seed are presented in Figs. 10, 11, 12 and 13.

The reduced population size and maximum allowed generations did not worsen the results and they still show the expected trend. The Pareto front is less populated as compared to that from polynomial approximation. The variation in α and β along the Pareto front is seen to be similar for both the cases. The slopes of best-fit line in $\omega - v_x$ plot are 0.8470 and 0.8418 for the sliding friction case and for the rolling friction case respectively.

It is also interesting to compare the results obtained through the polynomial approximation (PA) and through the actual simulation (AS). The performance metrics defined in the last section are calculated for these different cases and are presented in Table 1. This shows that even after reducing population size and total number of generations, the performance of the solution is not affected.

6. CONCLUSIONS

The dynamics of the snake-robot and the gait variables for optimum motion of the snake-robot is analyzed using an multi-objective evolution algorithm, NSGA-II. Two frictional cases are considered in the analysis and the results for both

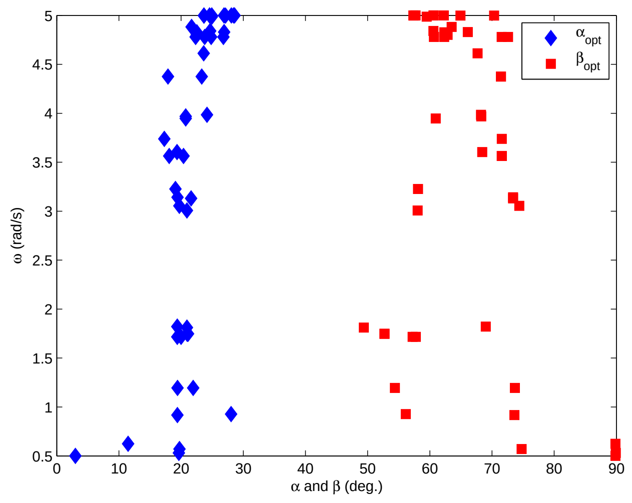

Figure 11: Design variables α and β along Pareto front using NSGA-II and actual simulation in the case of sliding friction. This shows that the optimum value of α increases slightly from 15 - 25 deg. and the optimum value of β remains constant around 60 deg. for ω varying from 0.5-5 rad/s.

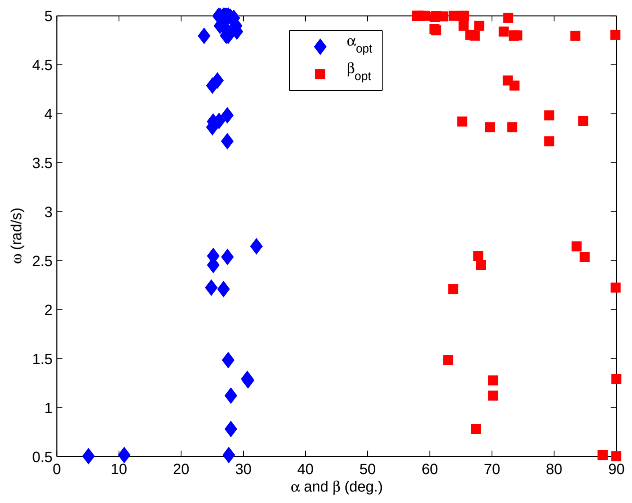

Figure 12: Design variables α and β along Pareto front using NSGA-II and actual simulation in the case of rolling friction. This shows that the optimum value of α remains constant around 25 deg. and the optimum value of β remains constant around 70 deg. for ω varying from 0.5-5 rad/s.

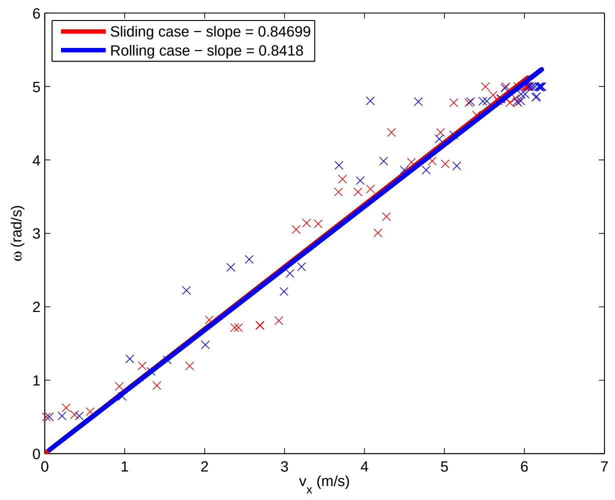

Figure 13: Design variable ω along Pareto front estimated using NSGA-II and actual simulation for both the frictional cases. The slope of the best-fit line is almost the same in two cases.

of them are presented. To speed up the numerical experimentation of the algorithm, an approximate method to find the objective function, called polynomial approximation, is implemented. A small parametrization study is conducted to find out a smaller population size and number of generations. Finally is algorithm is implemented for actual simulation of the objective function and results are shown.

In the polynomial approximation as well as in the actual simulation, the results are in good agreement with the literature for both the sliding friction model and for the rolling friction model. This not only confirms the results published earlier but also demonstrates the potential of NSGA-II for a fairly complex problem. It is worth to note that NSGA-II gave good results in spite the fact that the objective functions has scaling issues. The algorithm parameters suggested by the developers are found to be good starting points for the optimization process.

For large population and high number of generations, the algorithm gave results with a comparable accuracy with the graphical method. The parameterized population size and total number of generations are used for the optimization through the actual simulation. Though these numbers are considerably smaller than the earlier case, it is observed that the performance remains unaffected. Noting that the grid-search method took 9000 function evaluations while the NSGA-II based optimization took only 3840 function evaluations (approx. 57 % less), we feel that the objective of this study is fulfilled.

7. REFERENCES

[1] A. Belegundu and T. Chandrupatla. *Optimization Concepts and Applications in Engineering.* Prentice Hall, 1998.

[2] K. Deb. http://www.iitk.ac.in/kangal/index.shtml. website.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Algo.*, 6(2):187–192, 2002.

[4] K. J. Dowling. *Limbless Locomotion: Learning to Crawl with a Snake Robot.* PhD thesis, The Robotics Institute Carnegie Mellon University 5000 Forbes Avenue Pittsburgh, PA 15213, December 1997.

[5] C. K. Goh and K. C. Tan. Noise handling in evolutionary multi-objective optimization. In *IEEE Congress on Evolutionary Computation*, pages 1354–1361, 2006.

[6] S. Hirose. *Biologically inspired robots: snake-like locomotors and manipulators.* Oxford University Press, 1993.

[7] MATLAB. http://www.mathworks.com/access/helpdesk/help/ techdoc/ref/interp3.html. website.

[8] V. Mehta, S. Brennan, and F. Gandhi. Rigid-link snake robot dynamics and optimally efficient gait confirmed by experimentation. Submitted to IEEE Trans. Robotics Auto.

[9] P. Reed. System optimization using evolutionary algorithms. Lecture Notes, Spring 2007.

[10] M. Saito, M. Fukaya, and T. Iwasaki. Serpentine locomotion with robotic snakes. *Control System Magazine*, pages 64–81, 2002.

APPENDIX

A. C MODULES TO COMPILE NSGA-II IN MATLAB

The objective function is defined in following manner. Function ‘snakebot’ used here takes three inputs and spits two outputs. It can be changed to any valid MATLAB function name for any number of inputs and outputs.

```
void objfcn (double *xreal, double *xbin, int **gene, double
*obj, double *constr) {
mxArray *plhs, *prhs;
prhs = mxCreateDoubleMatrix(1, 3, mxREAL);
double *ptr = mxGetPr(prhs), *ptr1;
ptr[0] = xreal[0];
ptr[1] = xreal[1];
ptr[2] = xreal[2];
mexCallMATLAB(1, & plhs, 1, & prhs, "snakebot");
ptr1 = mxGetPr(plhs);
obj[0] = *ptr1;
obj[1] = *(ptr1+1);
mxDestroyArray(plhs);
mxDestroyArray(prhs);
}
```

The following subroutine is added to interface and compile the C-program with MATLAB. This subroutine requires one scaler value passed to the C-program, but it can be modified to pass no value or more values.

```
void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const
mxArray *prhs[]) {
double *x, *y;
int mrows, ncols;
if (nrhs != 1) mexErrMsgTxt("One input required.");
else if (nlhs > 1) mexErrMsgTxt(”Too many output argu-
ments”);
mrows = mxGetM(prhs[0]);
ncols = mxGetN(prhs[0]);
if (!mxIsDouble(prhs[0]) —— mxIsComplex(prhs[0]) ——
!(mrows == 1 & & ncols == 1))
mexErrMsgTxt(”Input must be a noncomplex scalar double.”);
plhs[0] = mxCreateDoubleMatrix(mrows,ncols, mxREAL);
x = mxGetPr(prhs[0]);
y = mxGetPr(plhs[0]);
nsga2r(y,x);
}
```