ME555 Project Final Report

Optimization on Snake Robot Performance

Prepared by:

Team 4

Fan Dong

Yixin Zhao

Prepared for:

A. Emrah Bayrak

Namwoo Kang

Alex Burnap

University of Michigan, Ann Arbor

Winter 2016

Contents

1. Background

Snake robot has been widely studied in the past several decades by lots of researchers. Because of its high flexibility, this kind of robot has advantages on working under some special environment comparing to other robots. Suitable working environment include: long tube, rubble, cave, and etc. Therefore, snake robot can be a good choice for research and rescue purposes under certain conditions. However, due to the limitation of its size, the capacity of the battery and motor were constrained. There is a need to study the balance between the size of the snake robot and its performance. In real world, snakes use many different gait patterns to move forward depending on the terrain and environment. In this project, we intended to optimize a snake robot that moves forward like a worm, which is called rectilinear gait.

Figure 1: A typical snake robot
(http://www.snakerobots.com)

The robot will move forward by first lifting up the last several modular. An arch will be formed during this process. The end of the last modular will then be moved forward and hold against the ground to provide the resistance, so that the arch can shift forward to the head of robot. This process will be repeated periodically to move the whole robot.

2. Optimization Objective

To obtain high performance, the key points for snake like robot is to increase its speed and the time that it can operate. Though it has high flexibility, the speed is limited by its geometry. Also, the operation time is an important issue to be evaluated, since we usually require the snake robot to work for a long time without recharge. In our design, we design batteries with same size in each modular. The batteries are connected in parallel. Therefore, the general optimization objectives for this system of snake robot are:
a) Maximize the possible operation time;
b) Maximize the speed.

Considering all these facts, we decided to optimize the snake robot from two perspectives for further analysis: physical layout and gait design. The whole system will be divided into two subsystems:

1) The physical layout includes the batteries and gearbox within one modular and the design of the geometry of the ‘snake’;
2) The gait design of the snake robot, which is to maximize the operation speed for a certain physical layout.
We plan to apply objective 1 for the first subsystem, which is to maximize the operation time and design objective 2 is applied to subsystem 2, which is to maximum the operating speed of the snake robot.

3. Design Variables

The whole snake robot is formed by connecting many segments, which is assumed to be a rectangular box in our model. And two adjacent segments are connected together by a flexible joint. Each segment will be actuated by a DC-motor, which is installed within each segment box.

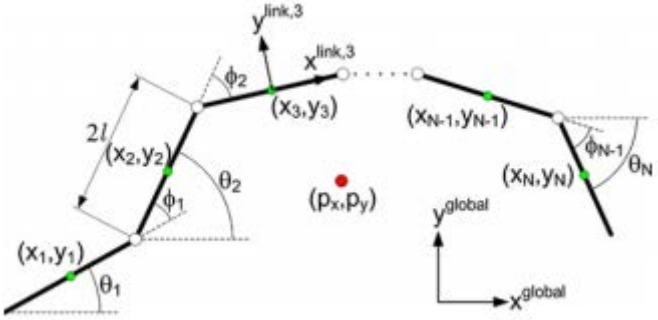

Figure 2: A geometric model of snake robot

Table 1: Variables used in snake model

Symbol	Description	Unit	Type	Applied sub-system
N	The number of links	-	Discrete	1
l	Half the length of a link (between two joint)	[m]	Continuous	1,2
a	The length of a segment	[m]	Continuous	1
b	The width/height of a segment	[m]	Continuous	1,2
d	Length of the Li-iron battery within the modular	[m]	Continuous	1
V_{motor}	Motor Volume	[L]	Continuous	2
φ	Gait angle	[rad]	Continuous	2

4. Sub-System Modeling and optimization

Fan Dong (00754182)

4.1 Subsystem 1 - Operation time analysis

Objective Function:

$$F_1 = -\frac{E_{total}(N,b,d)}{P_M(a,b,d,l,N)}$$

where E_{total} and P_M can be calculated as follow, detailed modeling shown in section 4.1.1:

$$\left(\frac{4}{3}(1-\cos(\phi_{\max}))+\frac{1.2\rho_m g}{18.903}\right)P_M$$
$$= (\rho_b b^2 d + 1.2 * \rho_{PVC} * (2b^2+4ab)t + 0.213\rho_m) g sin(\phi_{max}) * (M-1)(N-1) * v_0$$

$$E_{total}(N,b,d) = N * 230Wh/liter * b^2 d$$

Constrains: shown in detail in section 4.1.2.

Variables used in this subsystem analysis are shown below:

Table 2: Variables used in subsystem 1

Symbol	Description	Unit	Type	Applied sub-system
N	The number of links	-	Discrete	1
l	Half the length of a link (between two joint)	[m]	Continuous	1,2
a	The length of a segment	[m]	Continuous	1
b	The width/height of a segment	[m]	Continuous	1,2
d	Length of the Li-iron battery within the modular	[m]	Continuous	1

4.1.1 System modeling and objective function

In this part, we mainly concern the relationship between the required power and the capacity of the battery. Large battery energy storage capacity and smaller output power will provide longer operation time. In this snake robot, we simply assume batteries are installed in each segment

- Battery

Li-iron battery is widely used due to its high power density, long lifetimes and low cost. The volumetric power density of Li-iron battery is 230Wh/liter. We can simply calculate the capacity by:

$$E = 230Wh/liter * Volume$$

For convenient, we assume the battery have the same cross section geometry as the modular. In this way,

$$E = 230W/liter * b^2 d$$

All the batteries will be assumed to be connected in parallel, so that they can provide stable voltage to all motors. Therefore, total stored energy can be estimated by:

$$E_{total}(N, b, d) = N * 230Wh/liter * b^2 d$$

- Motor

We assumed the motor, gearbox and control circuit are embedded as the same modular. To get a relationship between the rated power of typical step DC-motor and its size, we investigated several step motor on the market and fit a model (Appendix A):

$$P_M[W] = 18.903 * V_m[L] + 3.3552 \quad \text{(i)}$$

Where V_m is the volume of the motor calculated in the unit of $liter$. In general, the motor has a length that is about 4-5cm, which will limit the maximum allowed number of modular number. Considering the size of battery, control circuit and gear box, we assume the number of modular has to satisfy: $N = 5,6,7,8$.

In this model, we do not care about the rotation speed and the rated torque, since they can be achieved by the gear box. From the required power P, we can calculate the volume of the motor (V_m). Due to the existence of other structure within modular, to be conservative, we estimate the total volume of the motor, gearbox and control circuit to by:

$$V = 1.2V_m \quad \text{(ii)}$$

- Mass Estimation

Each modular is mainly composed of the battery, motor, gearbox, control circuit, and the box. The density of the Li-iron battery is about $\rho_b = 2.55kg/L$. The modular of motor, gearbox and control circuit is assume to have a density of $\rho_m = 3kg/L$. The case of modular is assumed to be made of PVC, which has density of $\rho_{PVC} = 1.3kg/L$ We use following equation to estimate the mass for each modular:

$$m_0(a, b, d) = \rho_b b^2 d + \rho_m * V + 1.2 * \rho_{PVC} * (2b^2 + 4ab)*t \quad \text{(iii)}$$

Where t is the thickness of the box, which is assumed to be 5 mm. For convenience, the center of mass for each modular is assumed to be located at its geometric center.

- Power Requirement

The required power of the motor comes from the requirement that the motor should be able to lift up at most $M = N/2 - 1$ segments when N is even, and $M = (N - 1)/2$ when N is odd. To ensure the tail of the robot could touch the ground to provide friction, the minimum number of modular is N=5. In this part, we don't know how the robot will work. Therefore, we simply consider the gait angle is at its maximum achievable value: lift up M modular to a designated position (joint angle is at ϕ_{max}) within certain time (Figure 3).

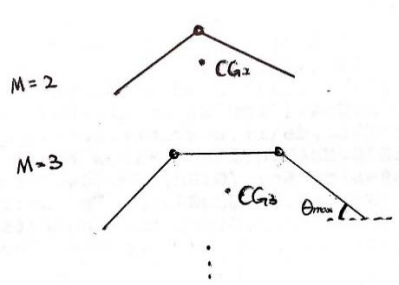

Figure 3: Geometrical representation for the lifted segments

The height of the center of mass for the lifted modular can be estimated by:

$$h_{Mcg} = \frac{2l * sin(\phi_{max})}{M}(M-1)$$

The following figure illustrate how the robot moves forward:

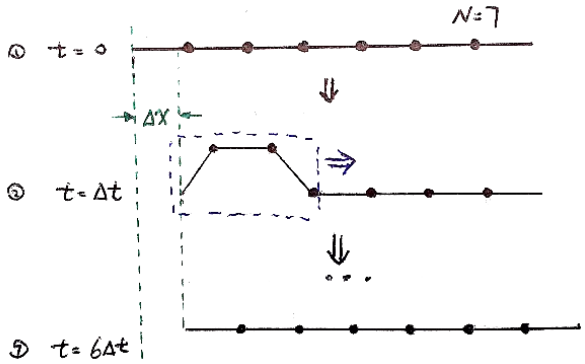

Figure 4: One possible gait pattern that will be used in this part of optimization

Therefore, the time that robot takes to move forward is $T = (N-1) * \Delta t$. The distance that the robot moves in one cycle is Δx, with:

$$\Delta x = 4l(1 - \cos(\phi_{\max}))$$

Since the speed of the robot is calculated in the gait analysis, thus we simply assume some velocity v_0 for the robot. Then, we can calculate the angular speed for the motor to lift up an 'arch'.

$$\omega = \frac{\phi_{\max}}{\Delta t} = \frac{\phi_{\max}}{T/(N-1)} = \frac{\phi_{\max}(N-1)}{\Delta x / v_0} = \frac{(N-1)\phi_{\max} v_0}{4l(1-\cos(\phi_{\max}))}$$

Since the torque and angular speed are varying during the lift process, we calculate the required motor power based on the rise of center of gravity. For convenience, we assume the power is constant during this process in this subsystem, later this velocity will come from subsystem 2. The robot velocity is assumed to be at 2 cm/s. To rise the center of gravity for the 'arch' segment to h_{Mcg}, the power that is required by the motor is given by:

$$P_0 = M * m_0 g * \frac{h_{Mcg}}{\Delta t} = \frac{m_0 g * sin(\phi_{max}) * (M-1)(N-1) * v_0}{2(1-\cos(\phi_{\max}))} \quad \text{(iv)}$$

To overcome friction and other unexpected disturbance, we assume the rated power for the motor $P_M = 2P_0$. Based on this power, we can calculate the required motor size, which will then affect the value of b and d. It turns out that the power and the mass for each modular are correlated, using equations (i)-(iv), we can get an expression for P_M:

$$\left(\frac{4}{3}(1-\cos(\phi_{\max})) + \frac{1.2\rho_m g}{18.903}\right) P_M$$
$$= (\rho_b b^2 d + 1.2 * \rho_{PVC} * (2b^2 + 4ab)t + 0.213\rho_m) g sin(\phi_{max}) * (M-1)(N-1) * v_0$$

So far, we can calculate the operation time for the robot (assume the robot keep working without rest).

$$T_{op} = E_{total}/P_M$$

Our objective in this part is to maximize the operation time, therefore, we set the objective function to be:

$$F_1 = -T_{op} = -\frac{E_{total}(N,b,d)}{P_M(a,b,d,l,N)}$$

4.1.2 Detailed Constrains in operation time analysis

From the primary design constrains described in part 4, detailed constrains in operation time analysis are listed as follow:

i) To ensure the robot has a proper length, the total length of the robot should be within the range of the desired length ($L_{total} = 1m$)

$$0.8 \cdot L_{total} \leq 2N \times l \leq 1.2 \cdot L_{total}$$

ii) Gait angle is at the maximum value that the robot can work at.

$$\varphi = arctan(\frac{2l-a}{b})$$

iii) Motor can be put inside the modular, so that the size of battery is limited

$$0 < d < f(P_0)$$

Where P_0 is the required power for the motor, and it can be used to determine the motor size.

iv) The length between two adjacent link should be within a certain range of the modular length, so that the modular won't look odd

$$1 < \frac{2l}{\mathrm{a}} \leq \frac{5}{4}$$

v) The width of the robot snake should not be larger than 1/20 of the total length (which is approximately the ratio a real snake would have)

$$0 < b \leq \frac{1}{20} L_{total}$$

4.1.3 Operation Time Optimization Results

Matlab is used to perform optimization analysis in this part. *fmincon* function was used to find the optimized point. More specifically, sequential quadratic programming (SQP) method was applied in *fmincon* for analysis. Detailed code is attached in Appendix C.

Since the number of modular N is discrete in this analysis, we evaluated the best result under N = 5, 6, 7 and 8. The following table shows the optimized results.

Table 3: Preliminary optimization result for operation time analysis

N	a [cm]	b [cm]	d [cm]	l [cm]	**t [hours]**
5	13.09	8.00	6.00	8.00	**12.55**
6	11.65	8.00	6.00	7.28	**12.26**
7	9.61	5.00	4.00	5.71	**2.88**
8	8.18	5.00	4.00	5.00	**2.88**

It turns out that the robot has a longest operation time when the number of modular is 5. Besides, the results show a pattern that the operation time are close when N/2 are rounded to the same integer. This is reasonable, since under these condition, the number of segments that are lifted in one cycle are the same.

4.2 Subsystem 2: Operation Speed Optimization

Yixin Zhao

4.2.1 Subsystem Objective Function and Problem Formulation

The optimization goal in this part is to find the optimal value of a certain number of gait parameters that maximize the forward velocity with some pre-determined physical parameters as constraints (obtained in Section 4.1, e.g. modular length, modular width and number of segments, etc.). Thus, the objective function to be minimized is

$$\min \qquad f = -v = -\frac{dx}{n \times \Delta t}$$

where n is the number of modulus, Δt is time required for the robot to finish one gait step (the process to transform from previous configuration to the next) and dx is the distance the robot travelled in one operation cycle (See Fig. 5 below).

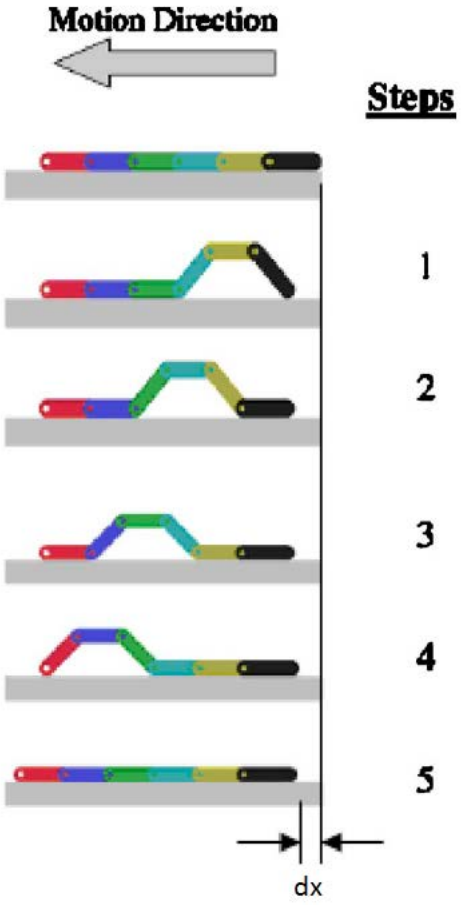

Figure 5: Snake robot rectilinear gait

Design variables and parameters will be discussed in detail in Section 4.2.2. Section 4.2.3 shows the detail of system modeling, analysis and problem formulation. Constraints will be discussed in Section 4.2.4 and finally the optimal results will be discussed in Section 4.2.5.

4.2.2 Design variables and parameters

For the speed optimization problem, there are four design variables and are listed in the table below.

Design Variable	Notation	Type
half module length,	l	continuous
module width,	b	continuous
gait angle,	φ	continuous
motor volume,	V_{motor}	continuous

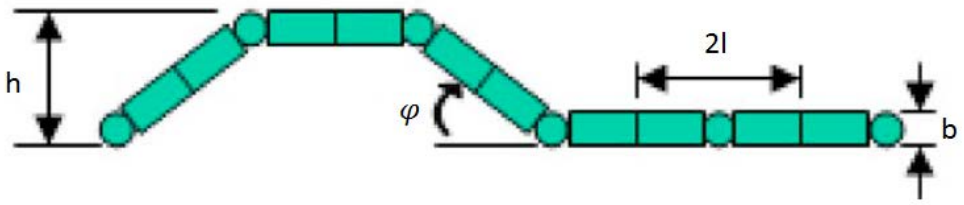

Figure 6: Design variables (Figure is retrieved from ref. [1])

There are two parameters in this sub-problem, number of modules n and the mass of the battery. The number of modules n is treated as a parameter since for one gait step only three modules are involved (as shown in the figure). Therefore, as long as $n > 3$, it does not affect the distance the robot will move forward in one complete gait cycle. However, it will increase the time required to complete a full gait cycle and thus if n is taken as a design variable, the optimizer will always try to set n as small as possible, which only has a monotonicity effect on the objective function and is irrelevant to the other design variables. Therefore we only consider it as a parameter obtained in the first sub-problem in this project.

4.2.3 System modeling

To evaluate the objective function, $f = -v = -\frac{dx}{n \times \Delta t}$, we need the expression of dx and Δt.

With the design variables defined, the forward distance dx for one gait cycle can be computed by comparing the original position of the last point of the robot with the position of the last point after the first gait step:

$$dx = 4l(1 - \cos\varphi)$$

To evaluate the objective function, we still need the expression of time Δt in terms of design variables and parameters. Δt is the time needed for the robot to complete one gait step and can be estimated by using the total amount of work needed on the joint divided by the power available from the motor, which is a linear relation with its volume as shown in Section 4.1.1.

$$P_{motor}[W] = 18.903 * V_{motor}[L] + 3.3552$$

To compute the torque and angular position on a joint is difficult since they are changing with time during

one gait cycle and also they are different on different joints for different configurations. In order to find the total work needed on a joint, we use pre-optimized gait trajectory (values of torque and angular position) as the normalized value. Then the total amount of work can be calculated by

$$W^* = \int \tau(t)\theta(t)dt$$

Ghanbari, Fakhrabadi and Rostami presented an optimal effort gait trajectory in 2009 and we use this model to calculate the work on the joint. It turned out that the largest amount of torque needed always occurs on the first joint of the lifted links as shown in Fig. 7.

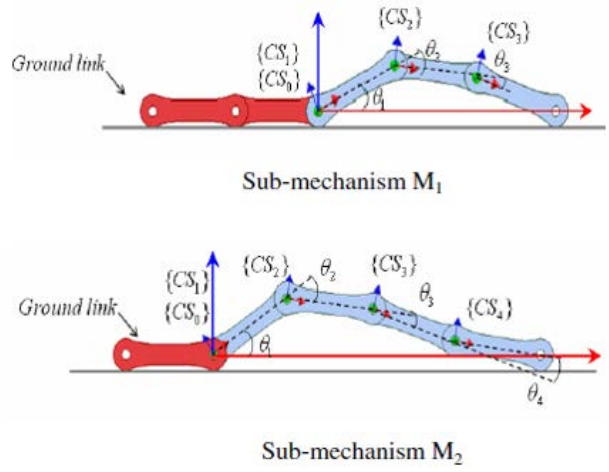

Figure 7: Largest torque needed always occurs on the first joint of the lifted links (marked blue)

Note from Figure 5 that there are four different configurations during one complete moving cycle, however, the third and fourth configurations are just the reverse process of the first and the second one so it's enough to only analyze these two configurations, which are shown in Figure 7. Also we only consider the joint which has the largest torque value. The optimal trajectory model from Ghanbari, Fakhrabadi and Rostami are shown below:

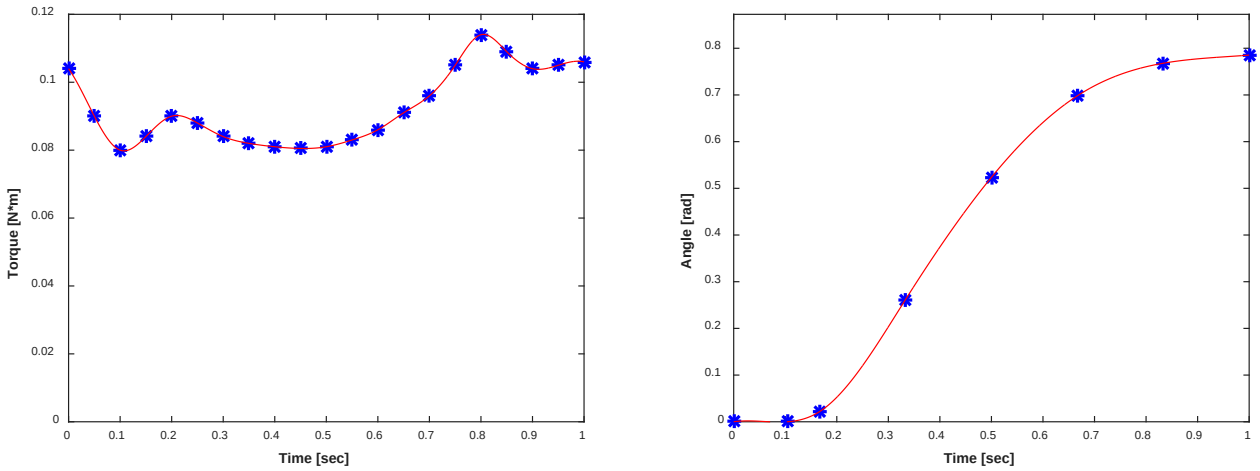

Figure 8: Normalized Optimal Gait Trajectory ($\tau(t)$ left; $\theta(t)$ right)

and we can calculate $W^* = \int_0^1 \tau(t)\theta(t)dt \cong 0.0428\ [J]$ and this will be the normalized value we use to calculate the work needed. The actual work needed on the joint is given by

$$W = \frac{1}{\Delta t} * \frac{\left(m_{battery} + \rho_{motor} V_{motor} + \rho_{PVC} l d^2\right)}{0.15} * \frac{l^2}{0.15^2} * \frac{16 * \varphi^2}{\pi^2} * \int_0^1 \tau(t)\theta(t)dt$$

where $m_{battery}$ is the parameter, $\rho_{motor} = 3kg/L$ is the average density of motor and $\rho_{PVC} = 1.3kg/L$ is the density of modular case.
Therefore, we have

$$\Delta t = \frac{W}{P_{motor}} = \frac{1}{\Delta t} * \frac{\left(m_{battery} + \rho_{motor} V_{motor} + \rho_{PVC} l b^2\right)}{0.15 * (18.903 * V_{motor} + 3.3552)} * \frac{l^2}{0.15^2} * \frac{16 * \varphi^2}{\pi^2} * \int_0^1 \tau(t)\theta(t)dt$$

The final expression of Δt is

$$\Delta t(l, d, \varphi, V_{motor}) = \sqrt{\frac{\left(m_{battery} + \rho_{motor} V_{motor} + \rho_{PVC} l b^2\right)}{0.15 * (18.903 * V_{motor} + 3.3552)} * \frac{l^2}{0.15^2} * \frac{16 * \varphi^2}{\pi^2} * \int_0^1 \tau(t)\theta(t)dt}$$

and the objective function can be written as

$$f = -v = \frac{4l(1 - \cos\varphi)}{n \times \Delta t(l, b, \varphi, V_{motor})}$$

4.2.4 Constraints for operation speed optimization

Gait design constraints are listed as follows:

- The maximum height that the robot will achieve during a complete gait cycle should also be constrained by the mission requirement. For example, if the environment only allows 20 cm above the robot, then the maximum height should not exceed 20 cm, which can be written as

  $$2l \sin\varphi - h \leq 0$$

  where h is some value specified by the mission requirement (See Figure 6).

- Gait angle φ could not exceed maximum angle allowed between two adjacent links, which is determined by geometry of the robot.

  $$tan\left(\frac{\varphi}{2}\right) - \frac{l}{5b} \leq 0$$

- The width of the modular should not be too small. In reality, motor and battery have some specified width and length, so the modular width should be bounded below. Also, we have observed that if we don't bound the width from below, the optimizer will always try to make the robot slimmer and longer, which has a positive effect on the speed. We set the lower bound to be 5 cm, which is approximately normal size of a motor with some margins.

  $$0.5 - b \leq 0$$

- The motor should be able to fit in the modular. Here we assume the motor and its accessories take at most half of the modular volume and the other half is for the battery.

  $$V_{motor} - b^2 \times 0.8l = 0$$

- There should be an upper bound for the motor power output, which is a linear function of motor volume. Since we are trying to maximizing speed, it's obvious that larger power input gives larger input. While in out model, the optimizer will try to make the motor size larger to achieve this. However, in reality, it's meaningless to have a robot that can operate very fast but only last for few minutes or seconds. Thus we put an upper bound for the power that can be used.

$$P(V_{motor}) - P(V_{power\ limit}) \leq 0$$

- The width of the robot snake should not be larger than 1/20 of the total length (which is approximately the ratio a real snake would have)

$$0 < b \leq \frac{1}{20} L_{total}$$

It turned out that when optimal is achieved, the first three constraints listed above are active and all of them come from geometry.

4.2.5 Subsystem Optimal Result

For operation speed optimization, we use $m_{battery} = 0.5\ kg,\ n = 6$.The optimizer use SQP to solve the problem. The optimal result is $\varphi = 25.84^{\circ},\ l = 5.735\ cm,\ b = 5\ cm$ and $V_{motor} = 114.7\ cm^3$. The optimal operation speed is $v = 3.474\ cm/s$.
Parametric study and sensitivity analysis will be discussed in the system level optimization.

5. System Level Optimization

5.1 System Level Objective Function and Subsystem Relation

The system level object function is chosen to be

$$f = \frac{1}{c \times \frac{1}{f_1} + (1-c) \times \frac{1}{f_2}}$$

where f_1 and f_2 are the objective function of the two subsystems and c is a weighting factor showing the relative importance of the two objectives.

For the system level optimization, the two subsystems are related to each other in such a way that, the objective value of subsystem 2, f_2 will be an input to subsystem 1 to evaluate the operation time, and number of segments and the size of the battery, which are design variables of subsystem 1, will be input to subsystem 2 to evaluate the speed. Furthermore, the two subsystems share two common design variables, modular length l and modular width b.

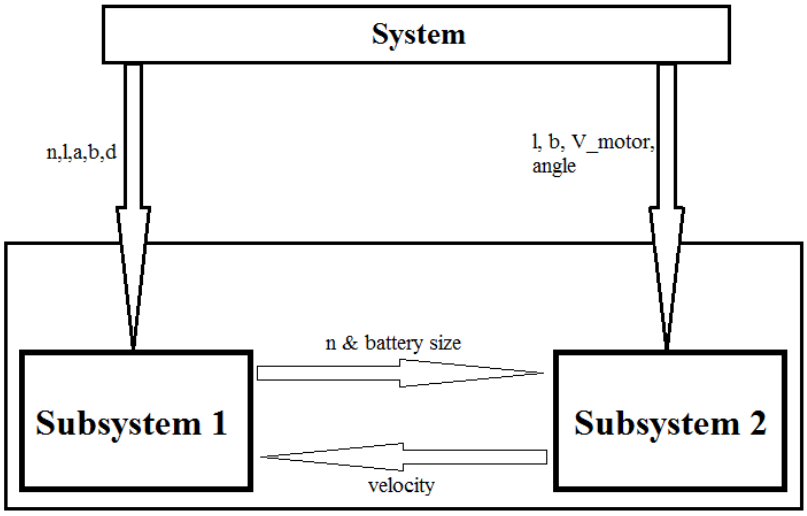

Figure 9: Subsystem relation flow chart

Figure 9 is a flow chart showing how the two subsystems work at the system level.

5.2 System-level design variables and constraints

System level design variables are listed below.

Symbol	Description	Unit	Type	Applied sub-system
N	The number of links	-	Discrete	1
l	Half the length of a link (between two joint)	[m]	Continuous	1,2
a	The length of a segment	[m]	Continuous	1
b	The width/height of a segment	[m]	Continuous	1,2
d	Length of the Li-iron battery within the modular	[m]	Continuous	1
V_{motor}	Motor Volume	[L]	Continuous	2
φ	Gait angle	[rad]	Continuous	2

In the above table, subsystem 1 is optimizing operation time and subsystem 2 is optimizing operating speed. There are two common design variables between the two subsystem, half module length and module width.

System level constraints are the same as subsystem constraints, which are listed below in detail:

- To ensure the robot has a proper length, the total length of the robot should be within the range of the desired length ($L_{total} = 1m$)
$$0.8 \cdot L_{total} \leq 2N \times l \leq 1.2 \cdot L_{total}$$
- Gait angle φ could not exceed maximum angle allowed between two adjacent links, which is determined by geometry of the robot.
$$\varphi - arctan(\frac{2l - a}{b}) \leq 0$$
- Motor can be put inside the modular, so that the size of battery is limited
$$0 < d < f(P_0)$$

where P_0 is the required power for the motor, and it can be used to determine the motor size.

- The length between two adjacent link should be within a certain range of the modular length, so that the modular won't look odd
$$1 < \frac{2l}{\mathrm{a}} \leq \frac{5}{4}$$
- The width of the robot snake should not be larger than 1/20 of the total length (which is approximately the ratio a real snake would have)
$$0 < b \leq \frac{1}{20} L_{total}$$
- The maximum height that the robot will achieve during a complete gait cycle should also be constrained by the mission requirement. For example, if the environment only allows 20 cm above the robot, then the maximum height should not exceed 20 cm, which can be written as
$$2l \sin\varphi - h \leq 0$$
where h is some value specified by the mission requirement (See Figure 6)
- The width of the modular should not be too small. In reality, motor and battery have some specified width and length, so the modular width should be bounded below. Also, we have observed that if we don't bound the width from below, the optimizer will always try to make the robot slimmer and longer, which has a positive effect on the speed. We set the lower bound to be 5 cm, which is approximately normal size of a motor with some margins.
$$0.5 - b \leq 0$$
- The motor should be able to fit in the modular. Here we assume the motor and its accessories take at most half of the modular volume and the other half is for the battery.
$$V_{motor} - b^2 \times 0.8l = 0$$
- There should be an upper bound for the motor power output, which is a linear function of motor volume. Since we are trying to maximizing speed, it's obvious that larger power input gives larger input. While in out model, the optimizer will try to make the motor size larger to achieve this. However, in reality, it's meaningless to have a robot that can operate very fast but only last for few minutes or seconds. Thus we put an upper bound for the power that can be used.

$$P(V_{motor}) - P(V_{power\ limit}) \leq 0$$

The above nine constraints are the same as the constraints in the subsystems. In addition to these nine constraints, there is one more constraint in the system level optimization problem:

- The motor volume must be the same for both subsystems, where in subsystem 2 the motor volume is a design variable, in subsystem 1 it is an intermediate variable used to calculate power and the values should be consistent, that is:

$$P_M - 18.903 * V_{motor} - 3.3552 = 0$$

Therefore, the system level problem has 10 constraints in total and 2 of them are equality constraints.

5.3 System level optimization results

We used sequential quadratic programming (SQP) method as the system level optimizer, which combined the advantages of many optimization algorithms and can provide results with high reliability. Since this method cannot deal with discrete problem, we analyzed the problem for all different N. To check if the optimality was achieved, we applied the optimizer for different initial points for all combinations of N and c that we would like to consider. The initial conditions for the optimizer was shown below:

Table 4: Initial evaluation points

a [dm]	b [dm]	d [dm]	l [dm]	N	ϕ [rad]	V_{mot} [dm^3]
10/N-0.4	0.6000	0.4000	5/N-0.1	N	1.0	1.0000
10/N-0.3	0.7000	0.5000	5/N	N	1.5	1.2000
10/N-0.5	0.5000	0.3000	5/N-0.2	N	0.5	0.8000

It turns out that the optimal solution points are the same for given c and N under all three initial points. Following table shows the optimal results:

Table 5: System level optimization results.

N	c	a [dm]	b [dm]	d [dm]	l [dm]	N	ϕ [rad]	V_{mot} [dm^3]	F	f1	f2
5	0.8	1.4100	0.5000	0.4316	0.8000	5	0.3632	0.1490	**-1.72**	**-36.912**	**-0.3573**
	0.9	1.2800	0.5000	0.4324	0.8000	5	0.3626	0.1496	**-3.29**	**-37.718**	**-0.3568**
	1	1.2800	0.6112	0.4506	0.8000	5	0.3505	0.1504	**-60.54**	**-60.540**	**-0.3000**
6	0.8	1.1100	0.5000	0.4563	0.6667	6	0.4200	0.1388	-1.70	-34.430	-0.3530
	0.9	1.1100	0.5000	0.4563	0.6667	6	0.4200	0.1388	-3.23	-34.430	-0.3530
	1	1.0667	0.6130	0.4681	0.6667	6	0.4103	0.1394	-53.89	-53.886	-0.3000
7	0.8	1.0402	0.5000	0.4000	0.6501	7	0.4796	0.0803	-1.58	-14.016	-0.3464
	0.9	1.0402	0.5000	0.4000	0.6501	7	0.4796	0.0803	-2.83	-14.016	-0.3464
	1	0.9143	0.5087	0.5316	0.5714	7	0.4223	0.0959	-22.97	-22.973	-0.3000
8	0.8	1.0402	0.5000	0.4000	0.6501	8	0.4796	0.0783	-1.41	-15.700	-0.3030
	0.9	1.0402	0.5000	0.4000	0.6501	8	0.4796	0.0783	-2.58	-15.700	-0.3030
	1	1.0289	0.5000	0.4117	0.6430	8	0.4751	0.0795	-16.35	-16.354	-0.3000

It turns out that for selected weight factor, $N = 5$ will always provide the best results. Comparing the results for $N = 5$ with the results in subsystem 1 and 2, we can see that all variables are different from the results in two subsystems. This is reasonable, since subsystem 1 tend to provide a longer operation time, thus will give a result with larger b and d value so that the battery can be larger. But larger b and d will restrict the gait angle that can be achieved in subsystem 2 and thus will reduce the working speed. In general, subsystem 1 and 2 tend to let the optimal variable values goes to two different directions. The system level objective function combined them together, and gives a balanced points between them. Therefore, the results in system level are different from both subsystem. When $c = 1$, we only consider subsystem 1. The results are still different from the result in subsystem 1. This is because of new constraints that are applied in this system level analysis.

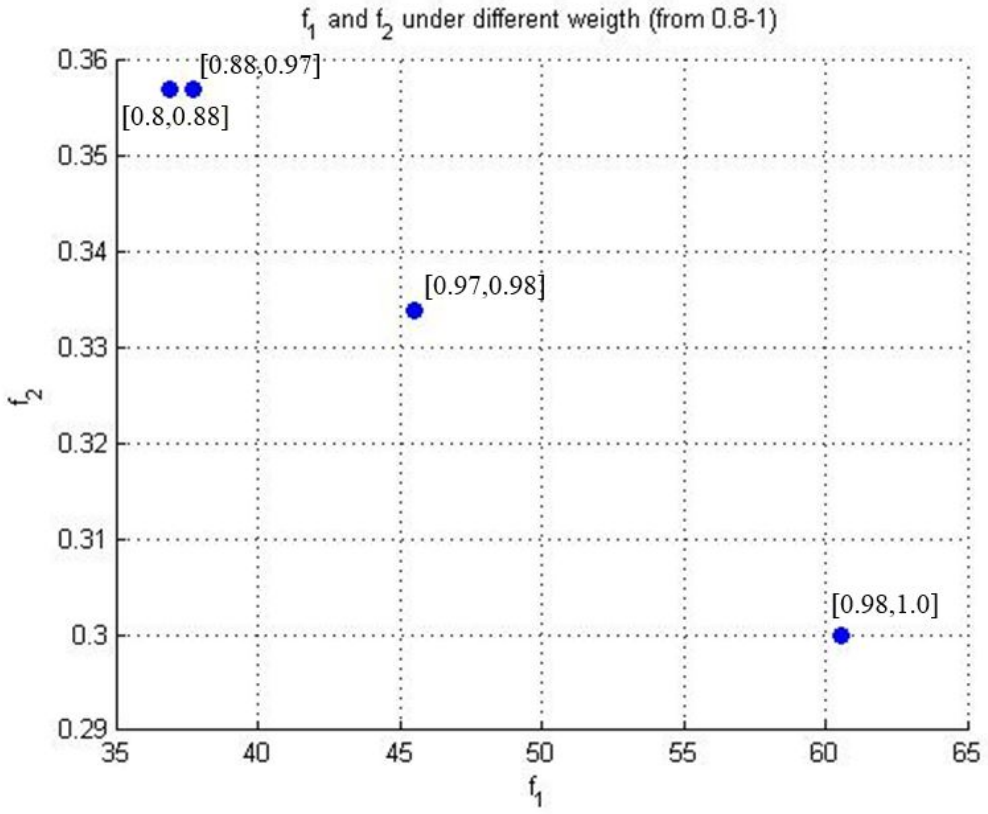

Figure 10: Pareto Front of System Level Optimization

Figure 10 shows the Pareto Front of the system level optimization. Instead of having a Pareto curve, we are getting four optimal points. Moreover, for weighting factor $c \in [0,0.8]$, the optimizer gives the same result as $c = 0.8$. The labels beside the optimal points in Figure 10 are the interval that weighting factor c lies within when the certain optimal is achieved. We are dealing with a hyper-dimensional design space, which cannot be visualized. We believe this is because the original system level objective function has at least four local minimum and one of them is the global minimum. As the weighting factor c changes, the shape of the function remains the same but the amplitude changes nonlinearly. This results in the global optimum to "jump" from one local minimum to another once the weighting factor passes some critical value. From Table 5, we see that for $c \in [0,0.8]$, the optimal design has an operation speed of 3.573 cm/s and can functioning for about 37 hours.

5.4. Active constraints

Since $N = 5$ will always provide best results, we want to evaluate the active constrains under this condition. The active constraints for $C = 0.8, 0.9$ and 1 are shown below.

- C = 0.8
  a.1) $b \geq 0.5$
  a.2) $l \geq 0.8$
  a.3) $2l * \sin(\varphi) + b - h \leq 0$
  a.4) $V_{motor} - 0.8d^2 l \leq 0$
  a.5) $\varphi - \arctan\left(\frac{2l-a}{b}\right) \leq 0$
- C = 0.9
  b.1) $a \geq \frac{8l_{min}}{5}$
  b.2) $b \geq 0.5$
  b.3) $l \geq 0.8$
  b.4) $2l * \sin(\varphi) + b - h \leq 0$
  b.5) $V_{motor} - 0.8d^2 l \leq 0$
- C = 1.0
  c.1) $a \geq \frac{8l_{min}}{5}$
  c.2) $l \geq 0.8$
  c.3) $2l * \sin(\varphi) + b - h \leq 0$
  c.4) $P(V_{motor}) - P\left(V_{power\ limit}\right) \leq 0$

6. Post Analysis

6.1. Parametric analysis

In this project, the only parameter that we can evaluate is the density of materials. Thus, we will evaluate the effect of mass change on the optimization results. Considering a density change on all robot components for $\pm 10\%$. For the case of N=5, results are shown below:

Table 6: Parametric analysis on density variation with N=5.

Density Variation	c	F		f1		f2	
		Results	%	Results	%	Results	%
0%	0.8	-1.72	N/A	-36.912	N/A	-0.357	N/A
	0.9	-3.29	N/A	-37.718	N/A	-0.357	N/A
	1	-60.54	N/A	-60.540	N/A	-0.300	N/A
10%	0.8	-1.70	-1.29	-33.896	-8.17	-0.354	-1.01
	0.9	-3.24	-1.61	-34.637	-8.169	-0.353	-1.009
	1	-55.69	-8.01	-55.691	-8.010	-0.297	-1.167
-10%	0.8	-1.74	1.34	-40.593	9.973	-0.361	1.04
	0.9	-3.34	1.68	-41.480	9.973	-0.361	1.037
	1	-66.46	9.77	-66.457	9.774	-0.304	1.233

We can observe from this table that f1 will change more than f2 when there is density change, which indicates that the operation time of the robot is more sensitivity to the change in material density. The effect of the density change will be attenuated in the system level objective function by the existence of subsystem 2 when the weight on subsystem 1 is smaller than 1.

6.2. Sensitivity analysis

From section 5.4, we know the active constrains for N=5 under C=0.8, 0.9, and 1. The sensitivity of these constrains can be calculated by evaluating $\partial f/\partial h$ at optimal points. We will consider the sensitivity of following constrains by evaluating the variation in f by applying a small perturbation on the active constraints at given optimal points.

- C = 0.8

Active constrains (a.1) and (a.2) were considered in this case.

Table 7. Sensitivity of active constraints (a.1) and (a.2) for N=5 and C=0.8

C	Constraint	∂h_i	F	F'	∂F	$\boldsymbol{\mu_{a.i}}$
0.8	(a.1)	01	-1.7200	-1.9036	0.1836	**1.836**
	(a.2)	0.1		-1.7642	0.0442	**0.442**
	(a.3)	0.05		-1.5586	0.1614	**3.228**
	(a.4)	0.05		-1.6814	0.0386	**0.772**
	(a.5)	0.05		-1.7199	1E-04	**0.002**

- C = 0.9

Active constrains (b.1), (b.2), and (b.3) were considered in this case.

Table 8. Sensitivity of active constraints (b.1), (b.2), and (b.3) for N=5 and C=0.9

C	Constraint	∂h_i	F	F'	∂F	$\boldsymbol{\mu_{b.i}}$
0.9	(b.1)	0.025	-3.2879	-3.2875	0.0004	**0.016**
	(b.2)	0.1		-3.5315	0.2436	**2.436**
	(b.3)	0.1		-3.3645	0.0766	**0.766**
	(b.4)	0.05		-3.0149	0.273	**5.460**
	(b.5)	0.05		-3.2262	0.0617	**1.234**

- C = 1.0

Active constrains (c.1) and (c.2) were considered in this case.

Table 9 Sensitivity of active constraints (c.1) and (c.2) for N=5 and C=1

C	Constraint	∂a	F	F'	∂F	$\boldsymbol{\mu_{c.i}}$
1.0	(c.1)	0.025	-60.5397	-60.0831	0.4566	**18.264**
	(c.2)	0.1		-58.6906	1.8491	**18.491**
	(c.3)	0.05		-63.7413	3.2016	**64.032**

References

Ghanbari, A., Fakhrabadi, M., & Rostami, A. (2009). Dynamics and GA-Based Optimization of Rectilinear Snake Robot. *ICIR, LNAI 5928*.

Hopkins, J., & Gupta, S. (2014, May). Design and Modeling of a New Drive System and Exaggerated Rectilinear-Gait for a Snake-Inspired Robot. *Journal of Mechanisms and Robotics*.

Hopkins, J., Spranklin, B., & Gupta, S. (2011, February 11). A Case Study in Optimization of Gait and Physical Parameters for a Snake-Inspired Robot Based on a Rectilinear Gait. *Journal of Mechanisms and Robotics*.

Appendix A: Motor analysis

Table A-1: Specification for some typical step DC-motor

Mode	Width [mm]	Height [mm]	Length [mm]	Volume [L]	V [V]	I [A]	Power [W]
39BYGL	39	39	34	0.051714	12	0.4	4.8
N/A	56.4	56.4	54	0.171772	3	2	6
57BYGH420-2	56	56	56	0.175616	3.2	2	6.4
42BYGHM809	42.3	42.3	48	0.085886	3	1.7	5.1
N/A	3	19	27	0.001539	3	1.2	3.6
N/A	42	42	34	0.059976	12	0.33	3.96
N/A	57	57	76	0.246924	3.08	2.8	8.624

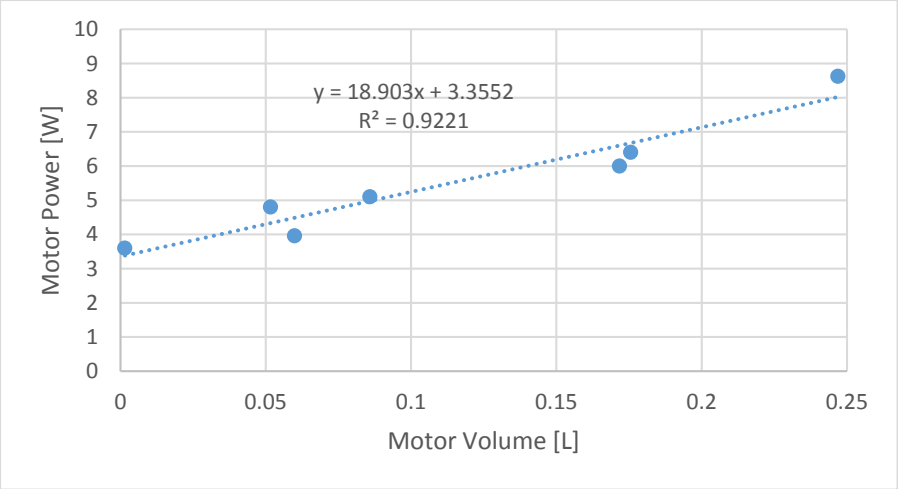

Figure A-1: Curve fitting for the relationship between motor power and size

Appendix B: System Level Optimization MATLAB Code

```
clear all; close all; clc;
% X(1) = a; X(2) = b; X(3) = d; X(4) = l; X(5) = N;
% X(6) = phi; X(7) = V_motor;
f = @(x) sys_obj(x);
% We need the number of design variables
% x(1)=a;  x(2)=b,  x(3)==d,  x(4)==l,  x(5)=N
N=6;                    % Number of Modular
if mod(N,2)==1            %Calculate M
   M=(N-1)/2;
else
   M=N/2-1;
end
num_variables =7;
x2_min = 0.5;               %b_min in [cm]
x2_max = 0.8;               %b_max in [cm]
x3_min=0.4;
x3_max=0.6;
x4_min = 0.08/(2*N)*100;     %l_min in [cm]
x4_max = 0.12/(2*N)*100;     %l_max in [cm]
x5_min = N;
x5_max =N;
x1_min = 8*x4_min/5;       % x_min based on l_min in [cm]
x1_max = 2*x4_max ;        % x_max based on l_max in [cm]

% Set the lower bound and upper bound
lb=[x1_min x2_min,x3_min,x4_min,x5_min,0,0]; % lower bounds of x1 and x2
ub=[x1_max x2_max,x3_max,x4_max,x5_max,2*pi,10e9]; % upper bounds of x1 and x2
x0=[100/N-4,6,4,50/N-1,N,1,1];                    % Set initial points for
optimization
A=[1 0 0 -2 0 0 0; -0.625 0 0 1 0 0 0; 0 1 0 -N*4/20 0 0 0;-0.6 0 1 0 0 0 0; 1 0.35
0 -2 0 0 0; 1 tan(20/180*pi) 0 -2 0 0 0];
b=[0; 0; 0; 0; 0; 0];              % A,b are used for unequal constrains Ax < b
Aeq=[0 0 0 0 1 0 0];
beq=[N];
% Aeq and beq are used for equal constrains Ax=b

OPTIONS = optimset('Algorithm','sqp');
[x,fval]=fmincon(f,x0,A,b,Aeq,beq,lb,ub,@sys_constr,OPTIONS)
```

```
function output = sys_obj(X)
% X(1) = a; X(2) = b; X(3) = d; X(4) = l; X(5) = N;
% X(6) = phi; X(7) = V_motor;

%Constants
g=9.81;
k=0.9;
ro_b = 2.55*k;
ro_pvc = 1.30*k;
ro_m = 3.00*k;
t = 0.05; % dm

% operation speed
% Parameters
m_battery = ro_b*X(2)^2*X(3);
k_motor = 3;rho_cha = 1.3;

% Output
del_t = sqrt(4.276E-2) *4/pi *X(6)/sqrt(18.903*X(7) + 3.3552)*sqrt((m_battery +
k_motor*X(7) + rho_cha*2*X(4)*X(2)^2)*4*X(4)^2/(0.15*1.5*1.5));
f2 = - 4*X(4)*(1 - cos(X(6)))/(X(5)*del_t);

% operation time
E  = 0.8*230*X(5)*X(2)^2*X(3)   %Total stored energy b,d are in cm
% Calculate
v_0 = 0.1*f2;
% v_0 = 0.2;
if mod(X(5),2)==1
   M=(X(5)-1)/2;
else
   M=X(5)/2-1;
end
phi_m = X(6);
A1=4/3*(1-cos(phi_m))+1.2*3*g/18.903;
A2=(ro_b*X(2)^2*X(3) + 1.2*ro_pvc*(2*X(2)^2+4*X(1)*X(2))*t + 0.212994*ro_m)...
   *g*sin(phi_m)*(M-1)*(X(5)-1)*2*v_0;
P_m=1.25*4*A2/A1;      %4motors works at the same time + 25% energy loss
f1 = E/P_m
f2

function [ C, Ceq ] = sys_constr(X)
```

```
% X(1) = a; X(2) = b; X(3) = d; X(4) = l; X(5) = N;
% X(6) = phi; X(7) = V_motor;

g = 9.81;
k = 0.9;
ro_b = 2.55*k;
ro_pvc = 1.30*k;
ro_m = 3.00*k;
t=0.05;


m_battery = ro_b*X(2)^2*X(3);
k_motor = 3;rho_cha = 1.3;

del_t = sqrt(4.276E-2) *4/pi *X(6)/sqrt(18.903*X(7) + 3.3552)*sqrt((m_battery +
k_motor*X(7) + rho_cha*2*X(4)*X(2)^2)*4*X(4)^2/(0.15*1.5*1.5));
v_0 = - 4*X(4)*(1 - cos(X(6)))/(X(5)*del_t);

if mod(X(5),2)==1
   M=(X(5)-1)/2;
else
   M=X(5)/2-1;
end
phi_m=X(6);
A1=4/3*(1-cos(phi_m))+1.2*3*g/18.903;
A2=(ro_b*X(2)^2*X(3) + 1.2*ro_pvc*(2*X(2)^2+4*X(1)*X(2))*t + 0.212994*ro_m)...
   *g*sin(phi_m)*(M-1)*(X(5)-1)*2*0.1*v_0;
Pm0=A2/A1;

C(1)=-X(2)^2*X(3)+(Pm0-3.3552)/18.903;
h = 1; % height limit; % dm
% geo constraint
C(2) = 2*X(4)*sin(X(6)) + X(2) - h;
C(3) = X(7) - X(4)*X(2)^2;
C(4) = X(6) - pi/2;
C(5) = 0.45-X(2);
C(6) = X(6) - atan((2*X(4)-X(1))/X(2));
C(7) = 0.3 - 4*X(4)*(1 - cos(X(6)))/(X(5)*del_t);
% power constraint
Ceq(1) = Pm0 - 18.903*X(7) + 3.3552;
end
```

Appendix C: Code for operation time optimization

```
function f=Timeobj(x)
%x(1)=a;    x(2)=b,    x(3)==d,    x(4)==l,    x(5)=N
% All length are in cm
% All density are kg/L

E    = 0.8*230*x(5)*x(2)^2*x(3)*10^-3        %Total stored energy b,d are in cm
%Calculate power
g=9.81;
ro_b=      2.55;
ro_pvc= 1.30;
ro_m=      3.00;
t=0.5;
v_0=0.2;
if mod(x(5),2)==1
      M=(x(5)-1)/2;
else
      M=x(5)/2-1;
end

phi_m=atan((2*x(4)-x(1))/x(2))
A1=4/3*(1-cos(phi_m))+1.2*3*g/18.903
A2=(ro_b*x(2)^2*x(3)*10^-3+1.2*ro_pvc*(2*x(2)^2+4*x(1)*x(2))*t*10^-3+0.212994*ro_m)...
      *g*sin(phi_m)*(M-1)*(x(5)-1)*v_0
P_m=2*A2/A1

f=-E/P_m;
```

```
clear; % Clear the workspace
close all; % Close all windows
%% EGGHOLDER FUNCTION -- ONLY COMMENT OUT ONE FUNCTION AT A TIME
f = @(x)Timeobj(x);

opt.alg = 'ga';

% We need the number of design variables
% x(1)=a;    x(2)=b,    x(3)==d,    x(4)==l,    x(5)=N
N=10;                                  % Number of Modular
if mod(N,2)==1                         %Calculate M
      M=(N-1)/2;
else
      M=N/2-1;
end

x2_min = 0;                            %b_min in [cm]
x2_max = 1/20*100;                     %b_max in [cm]
x4_min = 0.8/(2*N)*100;                %l_min in [cm]
x4_max = 1.2/(2*N)*100;                %l_max in [cm]
x5_min = N;
x5_max = N;
x1_min = 8*x4_min/5;                   % x_min based on l_min in [cm]
x1_max = 2*x4_max ;                    % x_max based on l_max in [cm]
% [x3_min,x3_max] = x3bd(x1_min,x2_min,x3_min,x4_min,x1_max,x2_max,x3_max,x4_max,N);    %d_min & d_max in
[cm] from other constrains

x3_min=0;
x3_max=2*0.6*x1_max;
num_variables =5;

% Set the lower bound and upper bound
lb=[x1_min x2_min,x3_min,x4_min,x5_min]; % lower bounds of x1 and x2
```

```
ub=[x1_max x2_max,x2_max,x4_max,x5_max]; % upper bounds of x1 and x2
x0=[11,3,6,10,N];                                % Set initial points for optimization
A=[1 0 0 -2 0;0 1 0 0 0;-0.6 0 1 0 0; 0 0 0 1 0; 0 0 0 0 1;1 0.35 0 -2 0];
b=[0;5; 0; 60/N; N;0];                          % A,b are used for unequal constrains Ax < b
Aeq=[0 0 0 0 0; 0 0 0 0 0; 0 0 0 0 0; 0 0 0 0 0;0 0 0 0 1];
beq=[0;0;0;0;N];
% Aeq and beq are used for equal constrains Ax=b

[x,fval]=fmincon(f,x0,A,b,Aeq,beq,lb,ub)
```