

University of Pennsylvania
Electrical & Systems Engineering Undergraduate Laboratories

ESE 112: Introduction to Electrical & Systems Engineering

Lab 6: EduBot Walking Gait Optimization

Sansern Somboonsong
Edited by Diana Palsetia
2008-2009

Objective

Provide students with a method of optimization for an unknown, multi-variable cost functions. Students will implement a gradient descent algorithm that will be used to tune the walking gait of the EduBot.

Background

Optimization is the study of problems with the objective to minimize or maximize a function often subjected to some constraints. There are many methods in solving optimization problems, each having its own purpose and weaknesses. It would take several graduate level courses to cover the prominent ones while new methods are being discovered and old ones improved upon by researchers, scientists, and engineers around the world. Optimization methods are commonly used in sciences, engineering, and finance. Improving the efficiency of systems is clearly essential and consequentially increases profitability.

In complex systems such as the EduBot, the dynamics of the subsystems are relatively well understood and can be individually analyzed. However, the dynamics of the overall system is difficult to analyze making it a challenge to devise a control scheme for the robot. In such situation, engineers often resort to other methods of optimization assuming that the cost function (the function to minimize) is unknown.

In this lab, students will write a program to optimize the walking gait of the EduBot with a goal to maximize the walking speed using a gradient descent algorithm. The cost function is the negative of the EduBot walking speed and can be minimized by changing 4 walking parameters described later in this document.

As an example, consider a single variable minimization problem with an unknown parabolic cost function as in Figure 1. If we have a known and differentiable function, we would simply take the derivative of $f(x)$, set it equal to zero, and solve for x . For an unknown cost function, a simple optimization algorithm would be to try an arbitrary initial value of the variable ($x = x_0$) and observe the value of the cost function $f(x_0)$, then incrementing x by a small increment dx obtaining $f(x_0+dx)$. If $f(x_0+dx) - f(x_0) > 0$, then

the function is sloping upward between x and x_0 and we need to move in the opposite direction of the increment to get to the minimum, by decrementing x_0 by some factor of the slope at x_0 (see Figure 1). Notice that if the increment is too large, we may never get to the actual minimum.

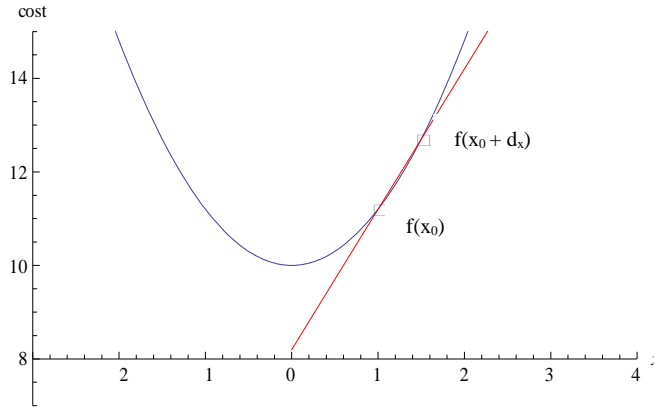


Figure 1 - Minimization Example 1

In the case of a multi-variable cost function, the cost function forms a surface in the n -dimensional space ($n-1$ representing the number of variables). For illustration purposes, we will consider a two-dimensional minimization problem. Although it is harder to visualize, the same concepts can be generalized for higher dimensions. To help us visualize the unknown cost function, assume it takes the shape of the surface shown in Figure 2. Similar to the one variable case, we start off with an arbitrary starting point $\{x_0, y_0\}$, find the change in the cost function (i.e. estimate the slope) in both x and y , and move in the opposite direction of the steepest slope. One can also imagine a box whose bottom takes the shape of the surface in Figure 2. If a ball is dropped into such box and the box is shaken, the ball is likely to roll or bounce to the lowest point of the box.

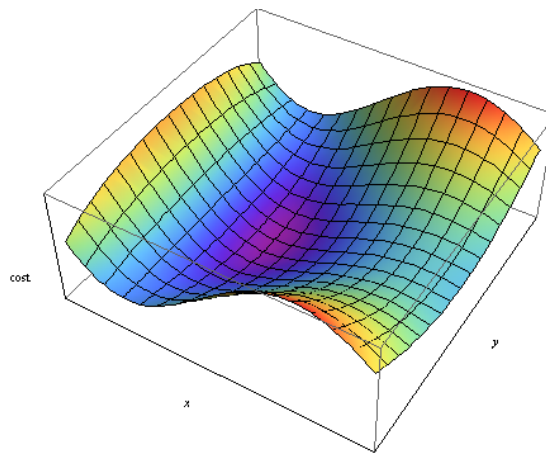


Figure 2 – Minimization Example 2

Gait Parameters

EduBot walks using the tripod gait described in the previous labs. Each leg has a fast and slow phase in a single cycle. The slow phase occurs while a leg is on the ground, and the fast phase occurs when the leg is in the air. Sets of three legs are synchronized to move together. You can convince yourself by observing how you walk and will observe that to catch yourself from falling over, the leg that is in the air must move forward fast and provide the support as the other leg ends its slow phase.

In the EduBot GUI's Manual Mode panel, several variables can be changed to alter the walking gait. We will however, concentrate on the four parameters.

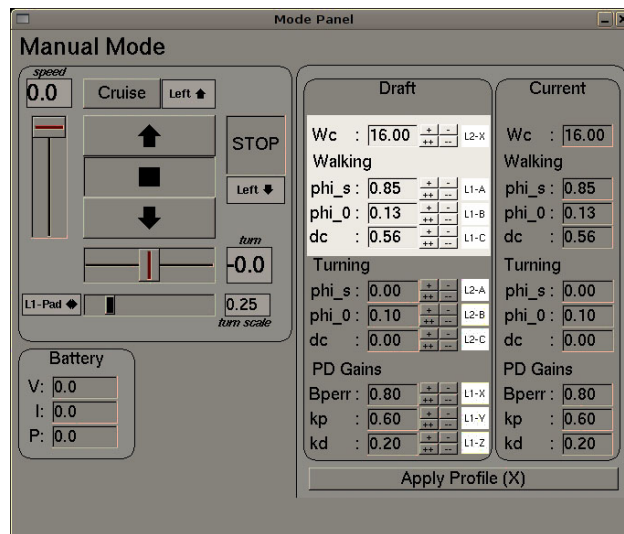


Figure 3 – Manual Mode Panel

Angular speed ($Wc = (10,18)$) is the speed in rad/sec of the leg rotation.

Sweep angle ($\phi_s = (0,\pi)$) is the angle in radians of the slow phase centered on ϕ_0 and will determine how large the slow phase is.

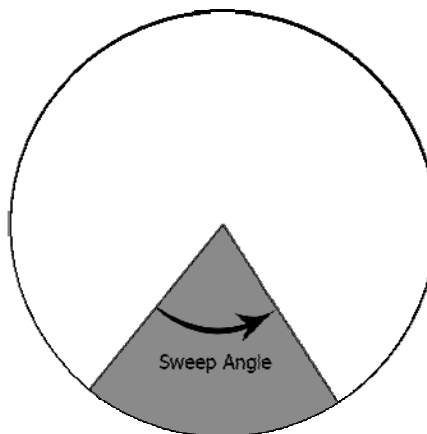


Figure 4 – Sweep angle

Duty factor ($dc = (0,1)$) is the percent of the time spent in the slow phase (on the ground). While the robot is walking, the duty factor should be more than or equal to 50%. While running or jogging, the legs are mostly in the air and the duty factor should be less than 50%.

Offset ($\phi_0 = (-\pi/2, \pi/2)$) determines where the center of the slow phase which in terms shifts the location of the slow phase. Changing this value can drastically alter the gait.

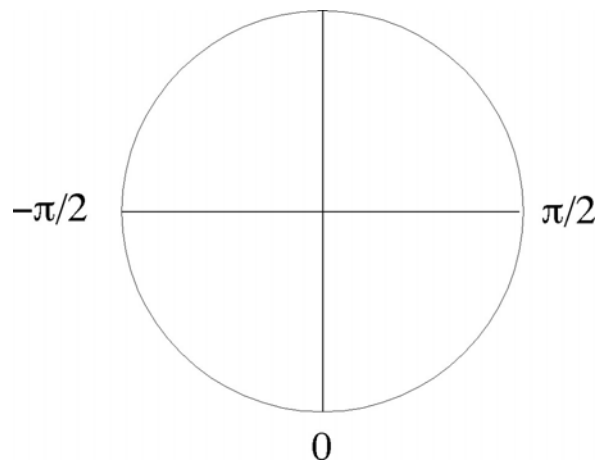


Figure 5 – Offset angle of slow phase

Note that the numbers in the brackets are constraints on the variables. Your optimization code should output a value within the given range.

Material

Computer with Linux Operating System
USB wireless adaptor
Joystick
EduBot
Stopwatch
Meter stick/tape measure

Prelab

1. Give one benefit of optimizing the robot's walking gait.
2. Define the following terms. Gait, optimization, and gradient descent.
3. Describe how you could use gradient descent optimization to increase the performance of the robot.
4. What potential problems are there in the basic gradient descent algorithm and how could they be overcome?

Lab Instructions

1. Write a stand-alone Java program to compute and implement gradient descent optimization for two variables. Both examples given prior have single minima. Make sure you account for multiple minima in your program. Your code should be general enough to optimize any function of two variables. It would however be useful to code the cost function in as one of your method. Part of your grade will be based on the sophistication of your code as well as your coding style. Some of the questions you might want to ask yourselves before you begin are:
 - a. How should the program handle multiple minima (local vs. global minima)?
 - b. How should the program know when to stop?

Your program should be able to minimize the following function:

$$f(x, y) = \sin\left(\frac{1}{2}x^2 - \frac{1}{4}y^2 + 3\right)\cos(2x + 1)$$
$$x \in (0, \pi/2)$$
$$y \in (0, 2\pi)$$

2. Use the previous program to optimize EduBot's walking gait for maximum "stable" velocity with two of the four parameters (choose the ones that you think will most affect the gait and explain why). Your goal is to maximize the speed of the robot, however the gait must be stable (be sure to include in your write-up how you defined stability).
As a reminder, you do not know the velocity of the EduBot as a function of the four variables and would also have to come up with a consistent way to measure the walking speed. The program should inform you of the parameter set to use, and allow you to enter the resultant velocity of the robot into your program.
3. For extra credit, you may alter your code to use all four parameters.

Program structure

You will be graded on your design and experimentation, so structuring your program from the start will save a lot time especially transitioning from part 1 to part 2. How you organize and work with data is also important, so think about the concepts learnt in ESE112 recitation or CIS110 or any equivalent Java course.

Program Requirements:

1. Must use Objected-Oriented style of programming (at least one object class must be implemented)
2. Must be stand alone i.e. must have main method (keep it separate from object class)
3. Can only use arrays as data structures.
4. Must have user input-output implemented for part 2. This will allow you setup initial parameter values, and input measured velocity for your program to evaluate the next set of parameter values (which will be inputted in the EduBot program).

Questions

1. Describe your implementation of gradient descent. Discuss any problems you encountered and how you overcame them.
2. What parameters did you choose and why?
3. How did you test your code to make sure it worked?
4. Did you need to make any changes to the basic program to get it to work for EduBot gait tuning?
5. What were the differences/difficulties that arose between optimizing some theoretical function and optimizing EduBot's gait?
6. Detail the results of the experiments with the robot (We want numbers and data must be tabulated).
7. How did you handle cases where the robot gait was unstable?
8. Describe your contribution to your group. As a percentage, how much of the work did you do? How much did everyone else do? Did your group work well together?

Submit your program to blackboard through the digital dropbox under the lab appropriate section (101/102) of Blackboard. Be sure to comment your code liberally so that the reader can easily understand the purpose of your code. Put your java file(s) in a folder called ese112_Optimization_XX_XX_XX (where XX is the first and last initial of a group member) and archive the contents. Finally submit ese112_Optimization_XX_XX_XX.zip to Blackboard.