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. Optimization methods are commonly used use in sciences, engineering, and finance.

Example, consider the following hypothetical problem:

- $x =$ sales price of Intel’s newest chip (in $1000’s of dollars)
- $F(x) =$ profit per chip (in $1000’s of dollars).
- Assume that Intel’s marketing research team has found that the profit per chip (as a function of $x$) is $F(x) = x^2 - x^3$
- Assume that we must have $x$ non-negative and no greater than 1

Objective function is profit $F(x)$ that needs to be maximized. Solution is to find the optimum chip sales price.

Using analytical method (using derivatives) we can solve for the optimum chip sale price as follows:

From calculus, the critical points of $F$, i.e. points of potential maximum or minimum are given by equation: $F'(x) = 0$ where $F'(x)$ is derivative of $F(x)$ with respect to $x$. For the hypothetical problem this would mean:

$F'(x) = 2x - 3x^2$

$2x - 3x^2 = 0$, we get the critical point when $x = 2/3$
Once the roots are found the sign of the second derivative tells if each of those points is a maximum or minimum. If sign is negative then we have found the maximum.

For our hypothetical problem the second derivative is:
$$F''(x) = 2-6x$$
When $$x = 2/3$$, $$F''(x) = -2$$. Since sign is negative, this is the maximum.
So, the optimal price per chip that maximizes the profit is $667.

Another example, where we can apply optimization problem is in complex systems such as the EduBot. In such a system, 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 or maximize) is unknown.

**Gradient Optimization**
A well known technique called gradient optimization is numerical method of finding minimum or maximum.

As an example, consider a single variable **minimization** problem with an unknown parabolic cost function as in Figure 1. 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). Thus take step proportional to the negative of the gradient (or the approximate gradient) of the function at the current point. Notice that if the increment is too large, we may never get to the actual minimum (because we would be oscillating around the minimum).

![Figure 1 - Minimization Example 1](image-url)
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](image)

In this lab, your goal is to 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 below.

**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.
Angular speed \((W_c = (10, 18))\) 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 \(\phi_0\) and will determine how large the slow phase is.

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 terns shifts the location of the slow phase. Changing this value can drastically alter the gait.
Note that the numbers in the brackets are constraints on the variables. Your optimization code should output a value within the given range.

Material

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

Prelab

1. Define the following terms. Gait, optimization, and gradient descent.
2. Give one benefit of optimizing the robot’s walking gait.
3. What potential problems are there in the basic gradient descent algorithm?
1. Write a stand-alone Java program to compute and implement gradient descent optimization for two variables. Example:

\[ f(x, y) = \sin\left(\frac{1}{2} x^2 - \frac{1}{4} y^2 + 3\right) \cos(2x + 1) \]

\[ x \in (0, \pi / 2) \]

\[ y \in (0, 2\pi) \]

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. Further see program requirements given below.

As a way to verify that your code is producing right results, we provide the minimum value and at what points it occurs i.e. \( f(1.0708, 2.8302) = -1 \).

Note that you can come across a minimum point but that may not the actual minimum. Therefore you must be able to account for global and local minimum. Some of the questions you might want to ask yourselves before you begin are:

a. What should be scaling factor (the value with which the slope is multiplied to get \( dx \))? Should this be constant?

b. How do you know the program has arrived at a minimum?

c. How should the program handle multiple minima (local vs. global minimum)?

d. How should the program know when to stop?

2. Note: if you start with a good program design, then you will require fewer changes to code from step 1. Part of your grade will be based on the sophistication of your code as well as your coding style.

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 ask you for the parameter set to use, and allow you to enter the resultant velocity of the robot into your program. Program should be able to provide the next set of values to try out on the robot.

3. For extra credit, optimize with 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 of 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 lecture.

Program Requirements:
1. Do not hard code things, except for the known function. Remember that your program must handle known as well as unknown function.

2. Must use Objected-Oriented style of programming (at least one object class must be implemented).

3. Must be stand alone i.e. must have main method (keep it separate from object class).
   - Main should create any objects, and have them do work or interact with other objects.
   - Should ask the user whether optimizing for known or unknown function.
   - Be able to handle inputs for part2 and provide results

4. For user input you use the built-in class Scanner. E.g. to take in user input of type double:

   ```java
   Scanner sc = new Scanner(System.in);//System.in implies from keyboard
double d = sc.nextDouble();  //reads in value as a double
   ```

   In order for you to create object of type Scanner, you will need to use import statement
   ```java
   import java.util.Scanner;
   ```

   When you execute the code, a box should appear in interactions pane to enter the value. If you enter a value other than double type then your program will crash.

5. Usage of arrays (objects and/or primitives).
   - Think of object that you write once and if you want to deal with many such object then we can use array of that object type. Hint: something that will help you to easily scale the number of parameters you need to optimize on.
   - Another place where you can use arrays is to hold past quantities. Hint: something to do with global and local minimum.
Questions

1. Describe your implementation of gradient descent. How did your program handle the potential problems?
2. How did you test your code to make sure it worked?
3. Did you need to make any changes to the basic program to get it to work for EduBot gait tuning?
4. What parameters did you choose to optimize the EduBot’s walking gait and why?
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 you 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.