Solving Continuous Nonlinear Programs with an Analog Computer: Final Report

Authors: Chong Andrew, Liao Thomas, Zou Alvin
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*— An analog computer that implements the Sequential Linear Programming (SLP) algorithm, which solves continuous Nonlinear Programming (NLP) problems. This work aims to solve these problems faster and more efficiently than current digital solutions. We demonstrate that our design can solve Nonlinear Model Predictive Control (NMPC) problems, which is a class of NLP problems. In particular, our analog computer swings up a damped inverted pendulum with optimal cost by solving a sequence of NMPC problems.**

***Index Terms*— Analog Circuit, Mathematical Optimization, Nonconvex Optimzation, Sequential Linear Programming**

1 INTRODUCTION

Traditional methods of solving continuous Nonlinear Programs (NLPs) rely on digital computation. We propose an analog implementation of the Sequential Linear Programming (SLP) algorithm [1], which is capable of solving NLPs at higher speeds and with lower energy consumption. Its principle of operation is demonstrated in Figure 1.

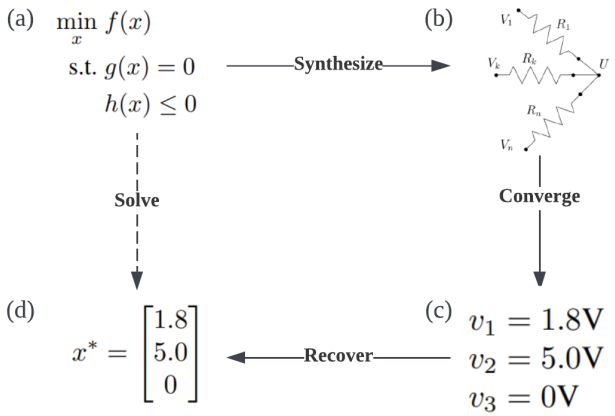

Figure 1: (a) NLP Problem (b) Analog Circuit [2] (c) Minimum-Energy State of the Analog Circuit (d) Solution to the NLP Problem

The scope of this project is to design and manufacture an analog computer that solves NLPs. We will extend upon the work of Dr. Sergey Vichik[1], who proposed a method of solving quadratic and linear programs using an analog circuit. We will provide a visual demonstration of the circuit solving NLPs. We do not aim to provide a theoretical account of the proposed method, manufacture hardware other than the analog computer, or provide compatibility to existing NLP solvers.

[1]MPC Lab, UC Berkeley

2 USE-CASE REQUIREMENTS

Based on our use case and project scope, we define the following requirements, split into functional requirements (FR) and non functional requirements (NR):

FR1 **Complexity:** The solver must be able to solve a NMPC problem with at least 9 variables and 3 constraints using the SLP algorithm

FR2 **Demonstrability:** The functionality of the solver should be demonstrated by a visualization

NR1 **Performance:** The solver should demonstrate a 10% increase in performance over at least one modern digital solver like Casadi

NR2 **Efficiency:** The solver should demonstrate a 10% increase in energy efficiency over at least one modern digital solver like Casadi

NR3 **Accuracy:** The solution produced by the system must be within ±10% of the optimal solution

NR4 **Cost:** The total cost of the project should not exceed $600

NR5 **Time:** The project should be completed within one semester.

3 ARCHITECTURE

3.1 Mathematical Background

The standard form of a continuous NLP is:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) = 0 \qquad \forall\, 0 \leq i \leq p \\ & h_i(\mathbf{x}) \leq 0 \qquad \forall\, 0 \leq i \leq q \end{aligned}$$

which consists of a cost function (a), an equality constraint (b), and an inequality constraint (c). In general, $\mathbf{x}$ is a vector, and g, and h are vector-valued functions, so the constraints in this formulation can be broken up into a system of multiple constraints.

We formulate the NMPC problem as an NLP:

$$\min_{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N,\mathbf{u}_0,\mathbf{u}_1,\ldots,\mathbf{u}_{N-1}} \left(\sum_{i=0}^{N-1} J_i(\mathbf{x}_i,\mathbf{u}_i)\right) + J_N(\mathbf{x}_N) \tag{1}$$

$$\text{s.t.} \quad g(\mathbf{x}_i,\mathbf{u}_i,\mathbf{x}_{i+1}) = 0 \;\; \forall\, 0 \le i \le N-1 \tag{2}$$

$$\underline{\mathbf{x}}_i \le \mathbf{x}_i \le \overline{\mathbf{x}}_i \qquad \forall\, 1 \le i \le N \tag{3}$$

$$\underline{\mathbf{u}}_i \le \mathbf{u}_i \le \overline{\mathbf{u}}_i \qquad \forall\, 0 \le i \le N-1 \tag{4}$$

The optimizer, given this problem, would try to minimize the cost function (1) we define, subject to dynamics constraints (2), state bounds (3), and control action bounds (4) by manipulating variables x_i and u_i.

The state variable x is defined as:

$$\mathbf{x} := \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

where θ_1 and θ_2 represent the angles of the two joints, while $\dot{\theta}_1$ and $\dot{\theta}_2$ represent the angular velocities of the two joints. x_0 is the initial state of the NMPC problem, given as a constant.

The cost function J is defined as:

$$J_i(\mathbf{x}_i,\mathbf{u}_i) = 0 \quad \forall\, 0 \le i \le N-1 \tag{5}$$

$$J_N(\mathbf{x}_N) = E_{\text{kin}}(\mathbf{x}_N) - E_{\text{pot}}(\mathbf{x}_N) \tag{6}$$

where E_{kin} is the kinetic energy of the system, and E_{pot} is its potential energy. The stage cost (5) is zero because we found it to be unnecessary for stabilizing the system. The terminal cost (6) penalizes kinetic energy and rewards potential energy, which encourages the controller to drive the double pendulum to the upright configuration and stabilize it there.

To discretize the continuous dynamics function of the double pendulum system, we employ the second order implicit Runge-Kutta method (IRK2). IRK2 has the advantage of achieving global error of $O\left((\Delta t)^2\right)$, while not adding intermediate variables. This is important because the number of components in the analog circuit is positively correlated to the number of variables. The main disadvantage of IRK2 is that there's no closed-form equation for the next state $\mathbf{x}_{t+1}$ given the current state $\mathbf{x}_t$ and control action $\mathbf{u}_t$, since it's an implicit method. This requires the equation to be solved by a rootfinder. [4][5] The integrated discrete dynamics under IRK2 satisfies:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \frac{1}{2}\Big(f(\mathbf{x}_t,\mathbf{u}_t) + f(\mathbf{x}_{t+1},\mathbf{u}_{t+1})\Big)$$

which translates into the dynamics constraint (2) as:

$$g(\mathbf{x}_i,\mathbf{u}_i,\mathbf{x}_{i+1}) = \mathbf{x}_{t+1} - \mathbf{x}_t - \frac{1}{2}\Big(f(\mathbf{x}_t,\mathbf{u}_t) + f(\mathbf{x}_{t+1},\mathbf{u}_{t+1})\Big)$$

We aim to solve the NLP formulation of NMPC with SLP [3], which operate as follows:

1. Generate local linear approximation of the original NLP at the current soluiton
2. Formulate a local LP subproblem based on the approximation
3. Solve the local LP subproblem with a NP solver
4. Update the solution of the original NLP
5. Go to step 1, unless the solution is within tolerance

More specifically, given an NLP:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) = 0 && \forall\, 0 \le i \le p \\ & h_i(\mathbf{x}) \le 0 && \forall\, 0 \le i \le q \end{aligned}$$

The local LP at step k is formulated as:

$$\begin{aligned} \min_{\mathbf{d}} \quad & \nabla f(\mathbf{x}_k)^\top \mathbf{d} \\ \text{s.t.} \quad & g(\mathbf{x}_k) + \nabla g(\mathbf{x}_k)^\top \mathbf{d} = 0 \\ & h(\mathbf{x}_k) + \nabla h(\mathbf{x}_k)^\top \mathbf{d} \le 0 \end{aligned}$$

The variables and the Lagrangian multipliers are then updated by:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}^*$$

where $\mathbf{d}^*$ is the optimal solution of the local LP [3].

The main idea of our approach is to solve local LPs quickly and efficiently with an analog circuit, which in turn makes SLP faster and more efficient.

3.2 Impracticalities of Vichik (2015)

3.2.1 Parasitic cost

While Vichik (2015) presents a way for one to solve QP problems, one needs to solve a linear programming (LP) problem in order to obtain the solve the QP. The current configuration using Vichik (2015) formulation does not allow for one to solve the optimization problem using SLP without significant changes.

In the design given by Vichick (2015), a LP need to be solved prior to the synthesis of the circuit in order to determine resistance values. This is impractical for our application because the QP circuit nees to by synthesized in real-tiem, and the extra time spent on

To compensate for this parasitic cost, this means that to solve a problem with SQP using the Vichik (2015) formulation, one would need to solve a LP problem for every step, which defeats the purpose of using a QP formulation. As a result, we rescoped our project to solve optimization problems using SLP.

3.2.2 Linear Cost Terms

The current formulation in Vichik (2015) is not able to fully express a quadratic cost function. This is because the current formulation only considers the quadratic terms in the cost function instead of both the linear and quadratic terms. As a result, this means that if the cost includes some linear cost, it cannot be expressed in the current formulation without requiring auxiliary variables or increasing circuit complexity.

3.2.3 Variable resistance resolution

The variable resistance resolution is lower than claimed. Vichik (2015) claims the following:

> (6.5) yields dynamic range of 102KΩ with resolution of roughly 3Ω that is equivalent to about 15 bit resolution.

However, doing this calculation ourselves shows that the resolution of the configurable resistance in the circuit is actually 12.61Ω instead of the claimed "roughly 3Ω". If this was taken as is, this would have resulted in a digital potentiometer that didn't have the required resistance resolution required to calibrate our circuit. As a result, the constraint circuit was redesigned to use two digital potentiometers at $1K\Omega$ and $100K\Omega$ to achieve 3.9Ω resolution. Furthermore, considering that this is quite a basic error, this meant that we went through the paper again confirming every detail to ensure that our design is correct.

3.2.4 OpAmp Rail Voltages

Vichik (2015) includes a proof of correctness of their equality constraint proof that implicitly assumes infinite OpAmp rail voltages. This is inconsistent with reality, where OpAmps cannot have infinite rail voltages and instead has a bounded rail voltage at a more reasonable value, such as 5V. If the circuit, while converging, reaches the rail voltages the circuit cannot converge to the correct solution due to the saturated OpAmps. Since it is impractical to increase the OpAmp voltages to an arbitrarily large amount, we resolved this issue by adding bounding constraints to each of the variables and reduced dynamic range from 100mV to 10mV at the cost of a higher noise floor.

3.2.5 Unbounded Problems

Vichik (2015) also does not account for problems in which the LP or QP solution is unbounded. Therefore, if an unbounded problem is given, the circuit cannot detect and handle these types of problems and instead outputs an invalid solution. To resolve this, we added added bounding constraints (like some modern software solvers do) and monitored OpAmp output voltages to ensure that they don't saturate.

3.2.6 U_{cost} Magnitude

Contrary to what the paper claims, increasing the magnitude of U_{cost} beyond U_{cost}^{crit} does not result in identical solutions. Instead, the accuracy of solution degrades if U_{cost} is more negative than some threshold.

> Theorem 3 will be proven in the following way. First, we show that there exists $U_{cost} = U_{cost}^{crit}$ such that any solution to (4.10) is also an LP solution; second, we show that for all $U_{cost} \leq U_{cost}^{crit}$ any solution to (4.10) is also an LP solution.

To solve this issue, we had to find a sweet spot for the value of U_{cost} by sweeping the voltage.

3.2.7 Mirror Variables

The paper does not consider the fact that only one of the variables x+ and x- needs to be connected to the digital potentiometers, due to the nature of the problem formulation. As a result, if we followed the circuit design presented in the paper we would have used 2x more digital potentiometers than what's actually required, which would've made the price of the circuit exceed our budget. We were able to solve this issue by multiplexing the digital potentiometers for each pair of mirrored variables with analog switches.

3.3 System Architecture

Our project is creating an analog computer that solves NLPs corresponding to a classical nonlinear control problem: swing up an inverted double pendulum. We plan to formulate the NLP using Nonlinear Model Predictive Control (NMPC). This implementation satisfies FR1 and FR2 because it solves a class of NLPs and is visually demonstratable. In addition, the resulting NLP contains a relatively small number of variables, and has a desirable sparsity pattern, which directly translates to a smaller analog circuit. This makes the scope of the project contained, and thus makes it easier to satisfy NR4 and NR5.

Our system contains two main components: the analog SLP solver and the demo platform.

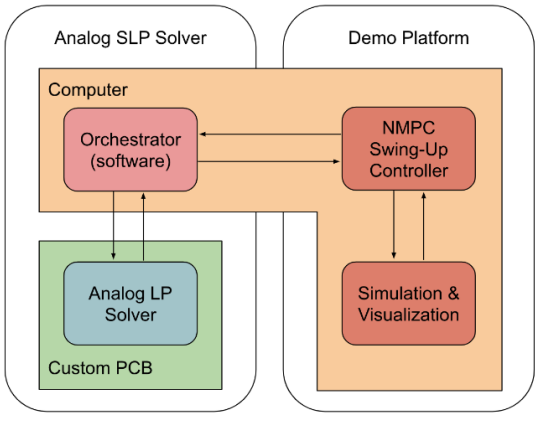

Figure 2: High level diagram of SLP System

3.4 Analog SLP Solver

The analog SLP solver consists of 2 components: the analog LP subroutine solver on a PCB, and a software orchestrator on a Raspberry Pi which implements the SLP algorithm by calling the analog LP subroutine repeatedly. Interfacing between the analog circuit and the software orchestrator will be done though analog-digital converters(ADCs), digital-analog converters (DACs), and programmable potentiometers.

3.4.1 Analog LP Solver

The analog LP subroutine solver is essentially a physical manifestation of the mathematical problem. It takes in a LP problem from the Raspi and solves it using the analog circuit. The optimal solution of the LP problem is equivalent to the lowest energy state of the circuit after it converges. The voltages are then read by the Raspi and converted back to numerical outputs. This process is illustrated in Figure 9.

The LP problem consists of three different components: the cost function, the equality constraints, and the inequality constraints. Each of these components can be synthesized using physical circuits, as illustrated in Figure 3.

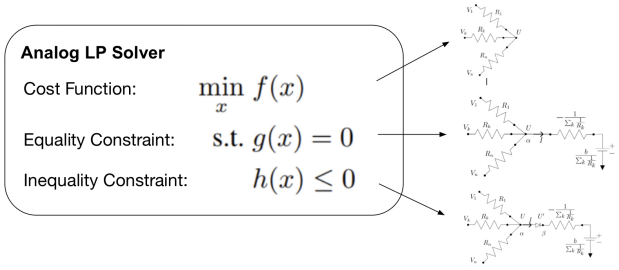

Figure 3: Circuits corresponding to components of the optimization problem

3.4.2 Equality Constraint

In order to realize the ideal equality constraint primitive, certain sections of the ideal circuit must be translated to represent its physical implementation. In particular, all variables require a linear potentiometer for the system to be tuneable and calibratable. Specifically for the equality constraint, the negative resistor is implemented using a TLE2037a op-amp, feedback resistors, as well as more linear potentiometers to calibrate. Using these components, the realized circuit can be constructed as shown in figure 5.

In figure 5, v_{01} and v_{12} represent input voltages for the equality constraint, followed by linear potentiometers to tune the input. The right-hand-side (RHS) of the circuit represents the negative resistor. Using this configuration, the effective negative resistance can be calculated by the following equation: $R_{eff} = -0.25R_p$, where R_p represents the total sum of resistances from the potentiometers and resistors $R79, R80, R81, R82$.

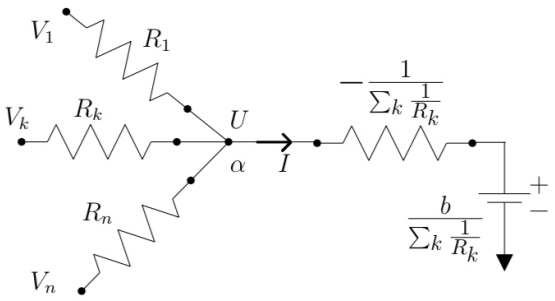

Figure 4: Ideal circuit Primitive for Equality

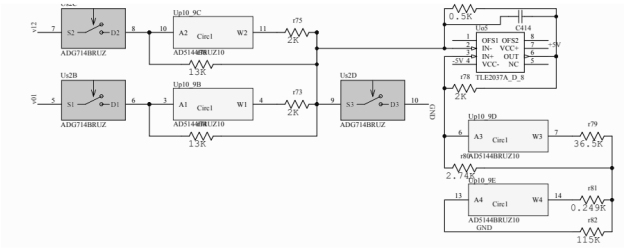

Figure 5: Realized circuit Primitive for Equality

3.4.3 Inequality Constraint

In order to realize the ideal inequality constraint primitive, certain sections of the ideal circuit must be translated to represent its physical implementation. The negative resistor can be implemented in the same method as above. Specifically, the circuit implements a single variable inequality constraint formulated as follows: $l_i \leq x_i \leq u_i$, where l represents the lower bounds, u represents the upper bounds, and x represents the optimization variable. As an ideal diode cannot be modeled with a standard diode, it is implemented using a switch and comparator. By using the comparator to measure the direction of current, it can either open or close the switch. The desired greater than or less than behavior can be modified by simply reversing the comparator input.

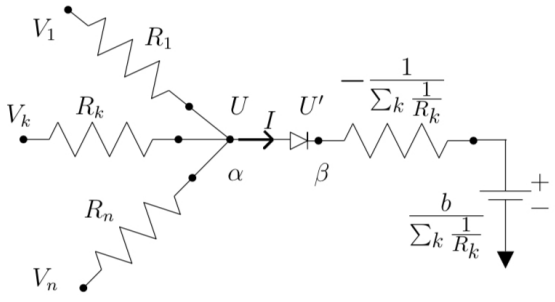

Figure 6: Ideal Circuit Primitive for Inequality

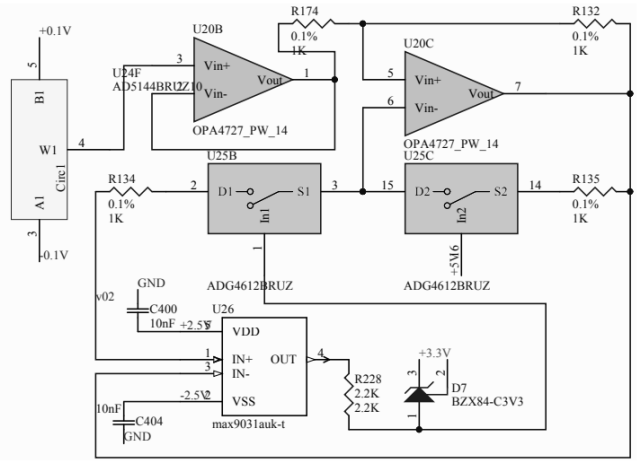

Figure 7: Realized circuit Primitive for Inequality

3.4.4 Linear Cost

Linear cost is simply a resistor that can be configurable for tuning, connected to a voltage source as shown in figure 8.

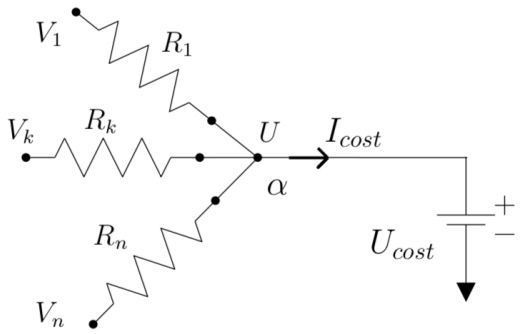

Figure 8: Ideal Circuit Primitive for Linear Cost

In order to realize the ideal linear cost primitive, this primitive is simply a linear potentiometer and switch, where the resistance is configurable using the linear potentiometer, as shown in figure 9.

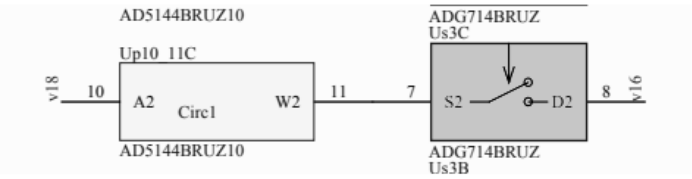

Figure 9: Realized Circuit Primitive for Linear Cost

3.5 Measurement System

The system aims to measure a voltage range of -5V to 5V for each variable in the optimal solution with an accuracy of at least 1mV.

3.5.1 Software Orchestrator

The software orchestrator will run the main loop of the SLP algorithm. It will call the analog LP solver multiple times with different inputs, as well as interfacing with the demo platform and providing the final solution to the controller after the solution converges.

3.6 Demo Platform

The demo platform consists of 2 components: the NMPC swing up controller and the simulation and visualization of the damped pendulum. The controller receives the solution from the analog SLP solver, translating them to torque controls that are then applied to the damped pendulum. The pendulum and its state evolution is then visualized.

4 DESIGN REQUIREMENTS

Based on our use case requirements and system architecture, we have defined the following design requirements(DR):

DR1 **PCB:** ADC/DAC used should be 16 bits

DR2 **PCB:** Total convergence time of the circuit should be under 28.8ms

DR3 **PCB:** The sample rate of the ADCs in the circuit should be at least 347kHz

DR4 **S-H Interface:** Communication between the PCB and the software orchestrator should be I2C or SPI

DR5 **S-H Interface:** Isolate LP subproblem from SLP main loop

DR6 **Demo Platform:** The user should be able to clearly see that a double pendulum is being swung up

DR7 **Demo Platform:** The speed of the visualization should be faster than 30fps

Where DR1, DR2, & DR3 are relevant to the PCB, DR4 & DR5 are related to the software hardware interface, and D6 & D7 are related to the demo platform.

4.1 DR1

Based on our research, 16 bits is the highest resolution we could find for ADC/DACs before they become excessively expensive. Additionally, 16 bits is a high enough resolution to convert between voltages and numerical values for our use case.

4.2 DR2

Based on our analysis of existing digital software solvers(e.g do-mpc), we found that the time it takes to solve the NMPC problem takes around 38ms, where 32ms is spent on solving the LP subproblem. Thus, to ensure our system satisfies NR1, the convergence time of the circuit should be under 28.8ms.

4.3 DR3

The time the software orchestrator has to wait for the circuit includes the convergence time and sampling rate of the circuit. To ensure the sampling rate does not introduce too much overhead into the system, we limit it to be below 10% of the convergence time.

4.4 DR4

To ensure we satisfy NR5, we should always simplify the design whenever possible. Thus, the communication between the PCB and the software orchestrator should be I2c or SPI. This is because these two protocols are robust for short distance transmission and are also widely used, which means the software orchestrator will likely have existing drivers in place, saving us the time to create them ourselves.

4.5 DR5

To be able to use the analog LP subroutine solver, the software orchestrator must be able to isolate the LP subproblem from the main loop of the SLP algorithm. It also requires an interface to provide the circuit with the inputs and convert them to voltages, as well as read the outputs voltages from the circuit and convert them back to numerical values. This step is critical to ensure the correctness of our solver(NR3).

4.6 DR6 & DR7

To satisfy FR2, the user should be able to clearly see that the pendulum is being swung up. This can be done through a live animation of the pendulum displayed on a monitor or a computer screen. The key requirements here is that the visualization should be large enough, as well as have a high enough frame rate so that the changes in the state of the system can be clearly captured.

4.7 Design Trade Studies

4.7.1 PCB Manufacturing/Assembly

There were many considerations in choosing a PCB manufacturing/assembly company to manufacture our product. This includes quality, turnaround time, ability to assemble the PCB components, and cost. We considered the following companies: JLCPCB, PCBWay, OshPark, and Colorado PCB Assembly.

It was determined that PCBWay was the best manufacturer for our use cases. This is because PCBWay generally has good quality, has the ability to both PCB manufacturing and assembly together, has turnaround + shipping in approximately a week, and offers large discounts if the assembly service is used. In fact, we found that if the assembly service is used, the price is actually cheaper than buying a stencil and manually reflowing it ourselves(saving us time and eliminating the possibility of manual errors). Some reasons why we didn't pick the other companies are as follows: JLCPCB has poor quality, OshPark can only manufacture the PCBs, and Colorado PCB Assembly, while with very high quality and fast turnaround, only offers assembly. Although it is possible to combine multiple options, we thought that handling the logistics with the component sourcing to different parts, as well as PCB shipping from the PCB manufacturer to the assembly and then back to us would take too much effort.

4.7.2 Spice Selection

To begin circuit simulation, a spice tool would be required to test the behavior of our preliminary circuits. Considering the wide variety of spice tools available, a detailed review of each of the available products was done.

We have decided on using LTSpice. This was because we realized the speed of learning and prototyping ability of LTSpice was much more important given our time constraints compared to the features available on the closest competitor, PSpice(by Cadence). Furthermore, the feature set that was available on LTSpice was sufficient for out needs. PSpice was highly considered considering its performance, as well as the fact that it is readily available to CMU students. Another contender was AltiumSpice, as our PCB design will be in Altium, so it would be easy to integrate. However, we decided against it considering its low performance.

5 SYSTEM IMPLEMENTATION

For our project, we have decided to use the rapid prototyping approach. This approach is suitable for projects where there are an abundance of unknowns to the team, which necessitates the team to learn while building. This applies to our project because analog computers for solving optimization problems is a relatively new topic pioneered by Dr. Sergey Vichik. By making multiple prototypes, we could extract technical insights in the process of building each of them, and learn our way towards the final product.

We have defined the following prototypes that incrementally build on each other and result in our final system.

- P1 A circuit simulation of a basic optimization problem
- P2 A circuit simulation of the analog LP solver
- P3 A visualized implementation of NMPC with a software LP solver in place of the analog LP solver

P4 Calculating the system dynamics for the inverted damped pendulum and visualizing the system

P5 Formulating the NLP of the pendulum swing up

P6 Converting the NLP to a form solvable by NMPC and isolate the LP subproblem

P7 A manufactured PCB that implements the analog LP solver

P8 A fully-integrated analog SLP solver with all hardware and software

The prototypes can be roughly split into two categories: the analog circuit and the remaining system.

5.1 Analog Circuit

The prototypes related to the analog circuit are P1, P2, and P7.

5.1.1 P1 & P2

While the conceptual primitives for equality, inequality, and cost are relatively simple, realizing their real-life counterparts are non-trivial. This is because the components in the conceptual primitives do not exist in real life, like an ideal diode or a negative resistor. As a result, they must be modeled by an equivalent circuit. Once these circuits are realized. P1 will create a simulation of a basic optimization circuit with each of the constraints. Once it is verified that the primitives work as intended, P2 will implement the LP circuit.

5.1.2 P7

After the analog circuit is validated in simulation, the schematic will be replicated in Altium and the PCB will be designed.

5.2 Remaining System

The prototypes related to the remaining system are P3, P4, P5, and P6.

5.2.1 P3

This prototype is to validate that double pendulum swing up can indeed be solved through NMPC using existing digital solvers. It ensures that our project is feasible before we put in too much time and resources, giving us the option to descope early in the project.

5.2.2 P4

For our project we need to isolate the LP subproblem from the SLP algorithm. Since most digital software solver libraries only provide high level solvers and does not give access to their low level functions, we need to create our own framework that allows for the isolation. The first step in this process is to calculate the dynamics of the double pendulum system and visualizing it. Thus, in P4 we create the basic framework for our solver and visualization.

5.2.3 P5 & P6

After verifying the feasibility of double pendulum swing up with NMPC in P4, the next step is to formulate the NLP of the system (P5). This prototype is the next step towards isolating the LP subproblem, and it expands on the framework created in P4, using the system dynamics in the problem formulation. Once P5 is achieved, we can then convert the NLP to a form solvable by NMPC and finally isolate the LP subproblem in P6.

5.3 Integration

After prototypes P1-P7 are all complete, we can integrate the different subsystems together to create our final product: an analog SLP solver that can demonstrate swinging up a double pendulum.

6 TEST & VALIDATION

Following our rapid prototyping approach, we have a similar testing strategy. We have devised a set of tests for our major prototypes, as outlined below.

T1 Run circuit simulation 30 times with random input voltages

T2 Run digital implementation 30 times and verify it solves the problem at least 90% of the time

T3 Run T1 using PCB

T4 Run T2 and T3 using the fully integrated solver

T1 tests for P1 and P2, T2 tests for P3, T3 tests for P7, and T4 tests for P8.

6.1 T1

T1 serves mainly two purposes. First, it validates that the analog circuit indeed solves the intended LP subproblem. This is tested by converting the output voltages of the circuit into numerical values and comparing them with an existing digital software solver. We need to ensure that the error margin is within 10% to satisfy NR3. Second, this test verifies that the circuit's performance meets NR1. This is done by calculating the time the circuit runs using a software timer and comparing the time with the digital solver. To ensure the robustness of this test, we will be running it 30 times with different input voltages. We are able to choose random inputs because we can feed the same inputs

into the digital solver and simply compare the outputs to confirm if the circuit is producing correct outputs.

We were able to execute this test. We did this by simulating the entire circuit in LTSpice and comparing the outputs against the reference solution.

6.2 T2

The main purpose of T2 is to validate that the damped pendulum swing up can indeed be solved by NMPC. It ensures that the problem we chose is solvable and demonstratable, thereby satisfying FR2.

We were able to execute this test. We made the solver swing up the pendulum in multiple starting configurations, and also tested it against adding noise to the position and velocity of the system. The solver was able to successfully swing up the pendulum in each of these cases. In addition, we compared the results of the SLP solver with another reference solver(IPOPT) and verfied the solution was correct.

6.3 T3

T3 is analogous to T1, except that instead of running on simulation it is run on the actual PCB board. To test this, the Raspi will send the inputs to the PCB, starting a timer simultaneously. The PCB receives the inputs and converts them to voltages, and runs the circuit, The Raspi actively monitors the PCB, and after detecting that the voltages converge, the Raspi will read the output voltages and convert them back to numerical values, and stops the timer. Apart from testing that the performance and accuracy both satisfy the requirements, we also test whether the power consumption satisfies NR2. This is done by connecting a power meter to the PCB and measuring the average power consumption after 30 runs.

Unfortunately, we were unable to perform this test due to delays from the manufacturer. As a result, we tried to test the entire system in simulation

6.4 T4

T4 is for testing our system after all subsystems have been fully integrated. Apart from running the previous tests in T2 and T3, for T4 we also evaluate the quality of the demonstration. We will measure if the frame rate of the demo satisfies DR7, and visually confirm that it satisfies DR6.

Since the PCB did not arrive in time, we decided to demonstrate our system purely using simulation. Since we have the circuit simulated in LTSpice, we created an interface between the simulated circuit and the rest of the software using PyLTSpice. Through this library, we could configure the configure the values of the resistors(potentiometers) and also recover the output voltages after the circuit converges. As such, we were able to integrate the circuit into our system and test the full system. We tested the system on three different starting configurations, and our system was able to solve all three problems correctly. Notably, in one configuration we placed a penalty on the torque, and our system was still able to find the optimal solution, which in this case was swinging the pendulum back and forth before finally swinging it up. As such, we are confident that our system works correctly.

We also measured the performance of our system. The results show that our system is able to achieve a 42% increase in speed on average while maintaining an error of less than 2.5% when compared against the digital solution.

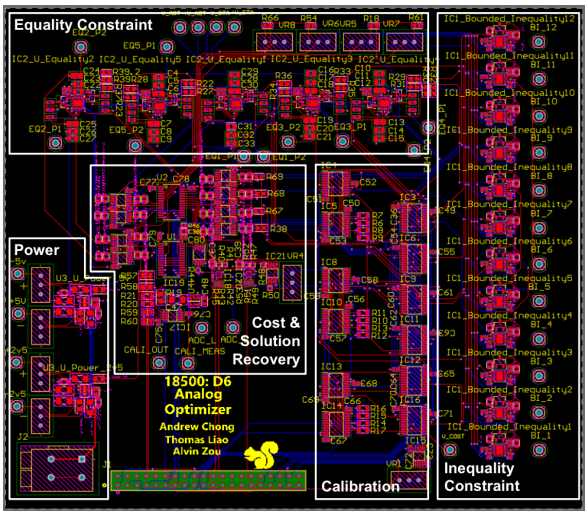

Figure 10: PCB layout of the analog LP circuit

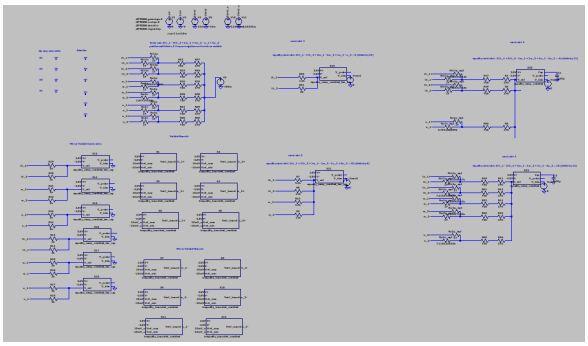

Figure 11: LTSpice simulation of the circuit

7 PROJECT MANAGEMENT

7.1 Schedule

Our project schedule is broken down into completing the prototypes we have defined in Section VIII. It is generally linear, expect that P2 depends on P6. This is because we can create the circuit simulation only after we have separated the LP subproblem, giving us the knowledge of what the cost function and constraints are. The gantt chart detailing our schedule is provided in figure 12.

7.2 Team Member Responsibilities

The breakdown of the team member responsibilities is as follows. Andrew Chong will be mainly working on the design and simulation of the circuit, as well as PCB fabrication. Alvin Zou will be mainly working on implementing the system dynamics and formulating the NMPC problem with Casadi. Thomas Liao will be working on the design and architecture of the system, as well as helping Andrew Chong with circuit design, and helping Alvin Zou with the SLP algorithm development and NMPC synthesis for the double pendulum swing up. All members will then work on the software hardware integration together. In addition, there is project manager that is rotated weekly among the members.

7.3 Bill of Materials and Budget

Bill of Materials			
COMPONENT	QTY	SUPPLIER	COST
OPA2328	10	Digikey	33.6
OPA2863A	23	Digikey	71.76
AD5144ABRUZ100	3	Digikey	24.54
AD5254BRUZ1	3	Digikey	25.77
ADS7056	2	Digikey	10.64
ADG706	1	Digikey	8.77
ADG714	5	Digikey	21.15
BAS70-04S	6	Digikey	3.24
LM4132AMF-2.5/NOPB	2	Digikey	8.32
LM27762DSSR	2	Digikey	3.94
PCB	1	PCBWAY	300.87
Raspberry Pi 5	1	Course Inventory	0
TOTAL			512.60

7.4 Risk Management

The tests defined in section IX will be tested on the major prototypes in our project. At the same time, there are major risks associated with these prototypes and so we have considered possible mitigation strategies in case the prototypes do not function as intended(the relevant test fails). The mitigation strategies for test failures are outlined below.

T1 Revise component specification and reselect components

T2 Retune NMPC parameters, such as increasing the prediction horizon and control horizon, and changing the cost function

T3 Reidentify parasitic effects and analyze possible manufacturing defects

T4 Reexamine the correctness of the software-hardware interface, such as the communication protocol and the conversion between voltages and numerical values

8 ETHICAL ISSUES

Our analog SLP solver can solve a specific optimization problem: swinging up a double pendulum. It's purpose is to solve a nonlinear optimization problem faster than existing digital solutions by using analog circuits. Our solver acts as a proof of concept, and its successful completion will indicate that other nonlinear optimization problems can be solved using a similar method with better performance and efficiency. This has potentially wide societal implications, largely due to the fact that many real world problems can be modeled as optimization problems. We list some examples below.

8.1 Health

Optimization helps make healthcare more efficient and accessible. For example, finding the most efficient distribution of vaccines during a pandemic can be formulated as an optimization problem. Solving this problem quickly can enable more effective control of infectious diseases.

8.2 Safety

Optimization helps make vehicles safer to operate. Vehicle control with safety guarantees is a classic application of model predictive control (MPC), which is an optimization problem. Our solver has the potential of solving MPC problems in real-time, which can make vehicle control strategies with stronger safety guarantees practical.

8.3 Welfare

Optimization helps distribute scarce resources efficiently. Logistics is another classic application of optimization, which is critical for fulfilling everyday needs, especially during periods of hardship. Solving these problems efficiently ensures that goods and services are delivered where they are most needed.

8.4 Economic Development

Optimization also has significant implications in terms of economic development. Specifically, this is because each of the systems relating to the production, distribution, and consumption of goods and services is an area that can be optimized to be more efficient, which leads to savings in terms of time and resources. Furthermore, optimization enhances the general productivity of our society by making production procedures faster, cleaner, safer, and more efficient. As a result, better optimization methods contributes to universal prosperity and sustainable development.

9 RELATED WORK

Our project build directly on the work done by Dr. Sergey Vichik [2]. We are extending his work by build-

ing a SLP solver that uses a analog circuit to solve the LP subproblem.

10 SUMMARY

In summary, we are building an analog SLP solver that can solve double pendulum swing up, which is a nonlinear optimization problem. It is designed to be faster and more energy efficient than existing digital solutions. We believe the major challenges for this project will be isolating the LP subproblem, designing the PCB, and integrating the subsystems into the final product.

10.1 Future work

For future work, we identified three directions to expand our project.

10.1.1 Using Sequential Quadratic Programming (SQP)

The first option is to Use SQP as the solver rather than SLP for better convergence and robustness. Doing so will enable the system to solve a much larger set of problems, as the performance of SQP is much better than SLP. To implement this change, the circuit must be modified to solve QPs rather than LPs.

10.1.2 Analog VLSI

The second option is to integrate the analog circuit onto a single chip with analog VLSI. This will increase the scalability of the system and enable the production of more complex circuits, which in turn can solve more complex problems.

10.1.3 Expanding generality

The third option is to extend the system to solve more general nonlinear optimization problems. This can possibly be done by modifying the circuit to solve more general forms of LPs/QPs.

10.2 Lessons Learned

Throughout this project we learned many lessons, here we list some of the most important ones below:

1 Verify the correctness of previous work before expanding it, especially when there's only a few paper written by the same person

2 Actively perform risk management, evaluating progress and planning for unknowns.

3 Rapid prototyping helps the team learn and identify problems early on in the design process

4 Decoupling the project enabled parallel development of the software and hardware, which made the transition of the solver design possible

5 Work at a maintainable pace to avoid stress, burnout, and mistakes

6 Share work and responsibilities between members, while maintaing good communication

Glossary of Acronyms

Include an alphabetized list of acronyms if you have lots of these included in your document. Otherwise define the acronyms inline.

- NLP - Nonlinear Program
- NMPC - Nonlinear Model Predictive Control
- SQP - Sequential Quadratic Programming
- QP - Quadratic Programming
- SLP - Sequential Linear Progremming
- LP - Linear Programming
- PCB - Printed Circuit Board
- RHS - Right Hand Side
- IPOPT - Interior Point OPTimizer

References

[1] Boggs PT, Tolle JW. "Sequential Quadratic Programming". Acta Numerica. 1995;4:1-51. doi:10.1017/S0962492900002518

[2] S. Vichik,"Quadratic and Linear Optimization with Analog Circuits". University of California, Berkeley, 2015.

[3] Nocedal, J., & Wright, S. J. (Eds.). (1999). Numerical optimization. New York, NY: Springer New York.

[4] Iserles, A. (2009). A first course in the numerical analysis of differential equations (No. 44). Cambridge university press.

[5] Süli, E., & Mayers, D. F. (2003). An introduction to numerical analysis. Cambridge university press.

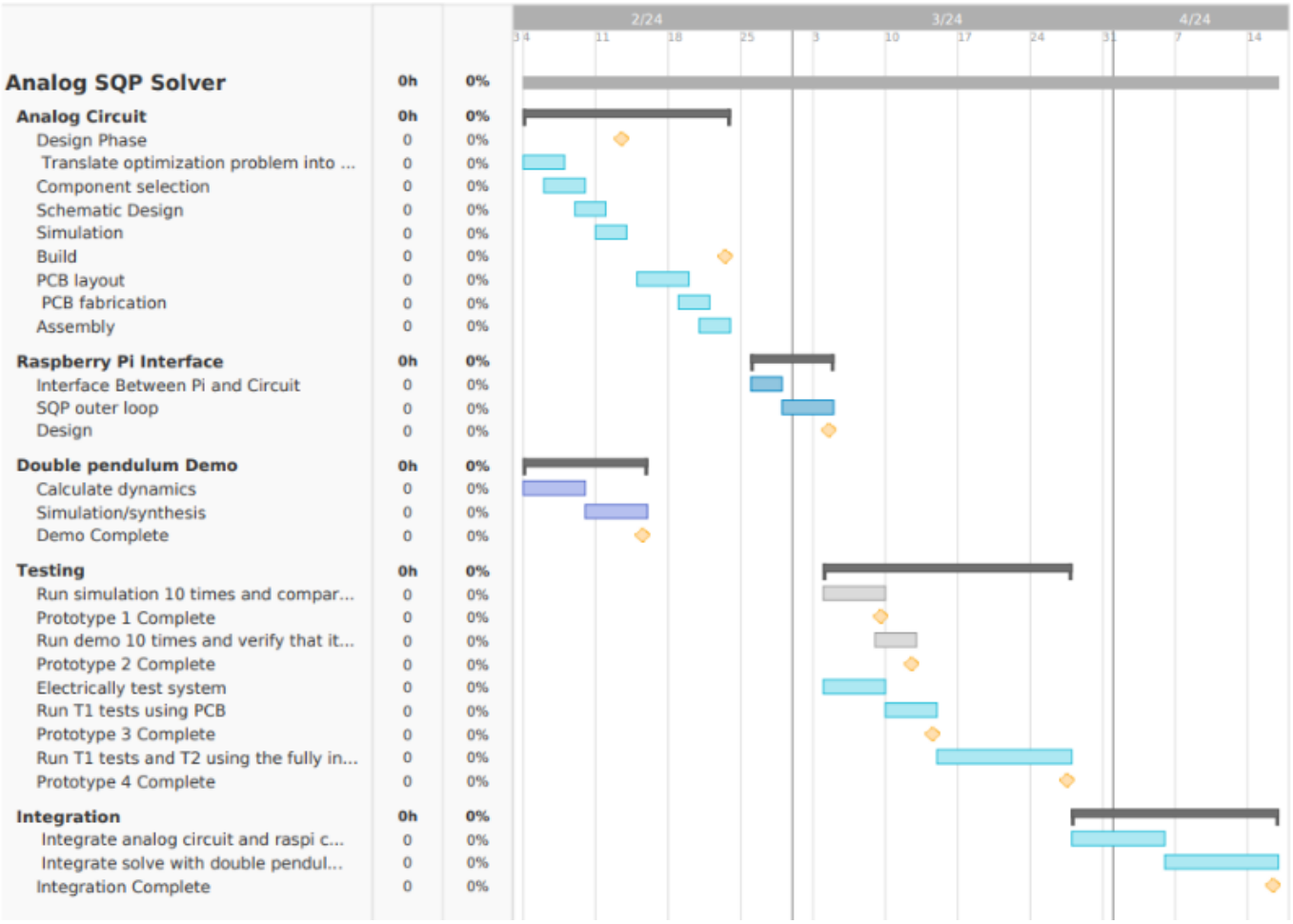

Figure 12: Gantt Chart