



Indian Institute of Technology, Bombay

DPAC: Digitally Programmable Analog Computer

Dhruv Shah, Sachin Goyal & Srivatsan Sridhar

Prof. Mukul Chandorkar
EE344: Electronics Design Lab, Spring 2018

April 2018

Abstract

Hardware-in-the-loop simulations are very commonly used to test controller design and monitor how the controller responds, in real time, to realistic virtual stimuli. In an HIL simulation, a real-time computer is used as a virtual representation of the plant model and a real version of the concerned controller. Most of these dynamical systems are in the form of coupled differential equations, and digital computers tend to be terribly slow at iteratively approximating solutions to such systems. The notion of using analog computing grids to efficiently solve differential equations (in hardware) has been well accepted in the research fraternity, and proves to be a faster way to solve linear dynamical systems.

In this project, we demonstrate a digitally programmable analog computer, which can solve linear dynamical systems with upto 5 state variables. The system is capable of working in real time, since there are no moving parts once the configuration is set and the system is programmed. The system is capable of being driven by upto 5 forcing functions, and can represent any linear dynamical system of the form

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx + Du\end{aligned}$$

It consists of active devices to implement integrators, gain blocks and inverter blocks using operational amplifiers, along with passive components to emulate the system matrix. These blocks will be linked together using analog switches which would be controlled by signals given by a microcontroller. For our first prototype, we assume $B = C = I_5$, for the sake of simplicity.

In this report, we present the design philosophies, layout descriptions, experimental results and analyses of two prototypes — DPAC- β and DPACv1.0. The DPAC- β is a miniature version of the DPACv1.0, to emulate second order systems, and features a block-modular structure and mechanical switches, allowing easy configuration of the system matrix and operational parameters. The DPACv1.0 features a single PCB, is interfaced and controlled using a microcontroller, and is capable of solving the linear dynamical system in real time.

Table of Contents

Abstract	2
Table of Contents	3
1 : Introduction	4
1.1 Objectives	4
1.2 Block Diagram	4
1.3 Background and Motivation	5
1.4 Target specifications	6
2 : System Overview	6
2.1 Design Alternatives	6
2.2 Required Subsystems	7
i) Power Supply	7
ii) Programmable Microcontroller	8
iii) Analog Computer Grid	8
3 : Project Implementation	9
3.1 Power Management Circuit	9
3.2 Microcontroller - MSP430 and MAX395 Switches	9
3.3 Using MSP430F168 for SPI Communication with Slave	11
3.4 Programming the MSP over USB	11
3.5 Active devices	12
4 : Performance Evaluation	13
4.1 First prototype: DPAC- β	13
4.1.1 Design	13
4.1.2 Experiments and Results	14
4.1.3 Observations & Inference	16
4.2 DPACv1.0	16
4.2.1 Features and Specifications	16
4.2.2 Testing and Evaluation of the Final Board	17
4.2.3 Experiments and Results	17
5: Conclusion & Future Work	22
APPENDIX	23
A: MSP430 Issues and Pt51	23
B: Setting up DPACv1.0 - A User's Guide	24

1 : Introduction

In this section, we begin by defining the objectives of the project, the motivation behind the choice, and the desired system specifications.

1.1 Objectives

- To build a single-board analog computer to solve linear dynamical systems described by an equation of the form $\dot{x} = Ax + Bu$.
- The analog computer will be programmed digitally, by programming an on-board FPGA/microcontroller using the computer.
- The analog computer will be made of basic blocks like integrator, inverter and gain blocks, interconnected with a dense layout of programmable switches.
- The microcontroller, once programmed will selectively open or close the switches to interconnect the blocks for the required circuit.

The final system will have an input to program the controller, input functions for the simulation and will output the response of the system being simulated.

1.2 Block Diagram

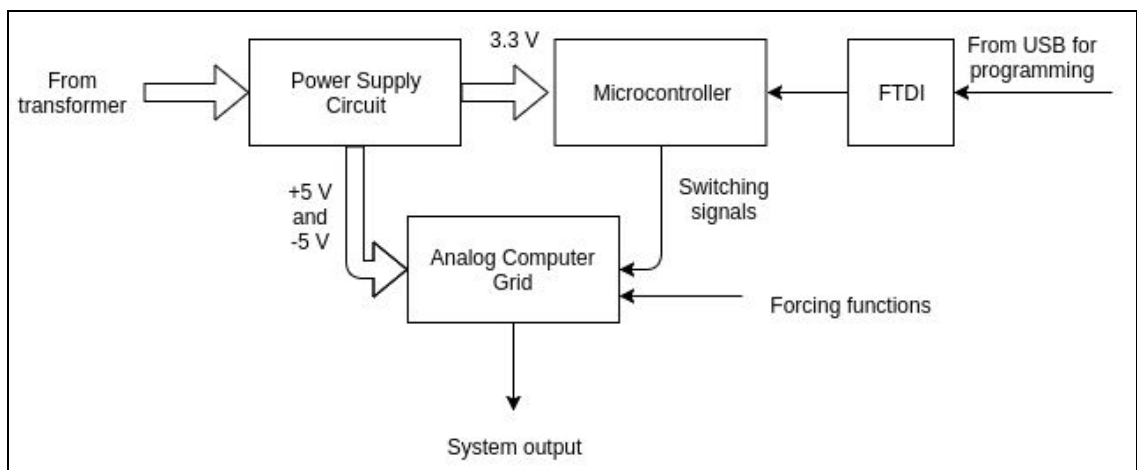


Figure: Block diagram of the proposed analog computer

1.3 Background and Motivation

Simulation of complex linear dynamical systems is important in the design and testing of any control system, to ensure that it works correctly on the given system even before applying it to the actual system. An arrangement called *hardware-in-the-loop* (shown in figure) is commonly used for this testing. Hardware-in-loop simulation is commonly used in testing of automobile control units where in-vehicle testing is time-consuming, expensive and not reproducible. This method involves virtually simulating the linear dynamical system (such as the engine of a car) on a computer.

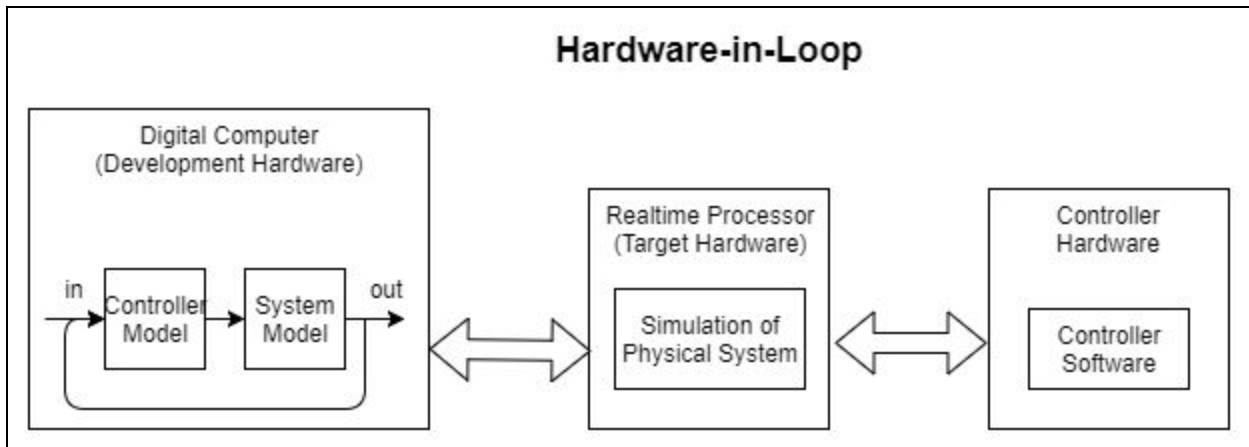


Figure: An illustration showing Hardware-in-loop simulation

Since a closed form solution of such systems does not exist, in practice, digital computers attempt to solve such systems by iteratively solving the differential equations involved. Not only is this time consuming, some higher order coupled differential systems may not even be feasibly solved on personal computers. An analog circuit that can model the required linear system can perform this simulation in real time, much like how the first analog computers worked.

While a digital process computes the answer through iterative steps, is time-consuming and computationally expensive, it assures the correct answer. An analog process works in real time but will offer limited accuracy in its answer. There is current research to study a hybrid approach using an analog process to quickly get an initial estimate and then using a digital process to get the precise answer.

The possibility of an analog computing attachment, that is digitally programmable, sounds very interesting, and has the capability to simulate real-time *hardware in loop* systems with great certainty, and at a much lower cost than employing computing clusters.

1.4 Target specifications

- Solve linear differential equations of the form $\dot{x} = Ax + Bu$. (We take $B = I$ for this system. This simplifies the prototyping phase, and can always be extended to a generic B matrix using gain blocks)
- The state variable vector x may have at most 5 elements.
- There may be at most 5 forcing functions in the vector $u(t)$
- The coefficients in the matrices A and B can be signed and allow easy reconfiguration by the user.
- The system must be self-sufficient, with onboard power management and microcontroller interfacing.

2 : System Overview

2.1 Design Alternatives

The figure below illustrates the possible pathways of designing an analog computing grid. The first approach, using configurable analog blocks (CABs), is inspired from the idea of using look-up tables (LUTs) in FPGAs - using a common building block of analog components, which can then be used as either an integrator, gain blocks, and so on. Such an arrangement increases the versatility of the usage and builds towards the notion of a Field Programmable Analog Array (FPAA).

The second design approach would be to make dedicated blocks for each task - gain blocks, summers, integrators and inverters. This makes the whole system modular and easier to interpret and implement. The fact that each block is now a dedicated one also reduces the complexity of each block, at the cost of designing more blocks. The grid design can be further divided into two approaches - (i) using a fully-connected, fixed grid, which allows the possibility of representing dense matrices A and B , and (ii) using suitable routing algorithms that only invoke as many blocks as would be required by the dynamical system to be solved. This may reduce the number of components required, but would increase the computational complexity of the routing algorithm.

Using configurable blocks to represent the system would require a much larger number of passive elements, a large fraction of which would remain unused or idle. The routing

complexity would also be quite significant, and hence, this approach sounds overkill for the system that we are aiming to model. Using dedicated blocks presents a modular architecture and better usability of the analog components that form the grid.

We will be attempting to solve the problem by this approach of using 3 levels of dedicated blocks to compute the intermediate variables. A good way to begin would be to use suitable routing algorithms, controlled by low resistance analog switches, which are controlled using an FPGA/microcontroller, to create a graph of connections in the analog grid.

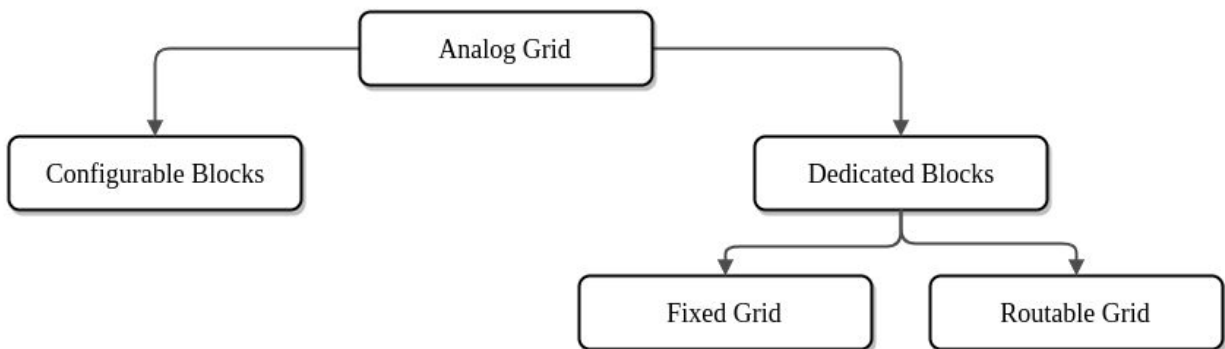


Figure: Design paradigms for implementing an analog computational grid

2.2 Required Subsystems

This section is a short description of the major subsystems that would be required in this project. Implementation details of the subsystems will be explained in the next section.

i) Power Supply

In our design, the microcontroller has a voltage level of 3.3V. We have used analog switches and op-amps which have a bipolar supply of +5V and -5V. We derive our power supply from 9V AC taken from a 50Hz transformer. Thus, we require a circuit to supply 3.3V, +5V and -5V from the 9V AC voltage.

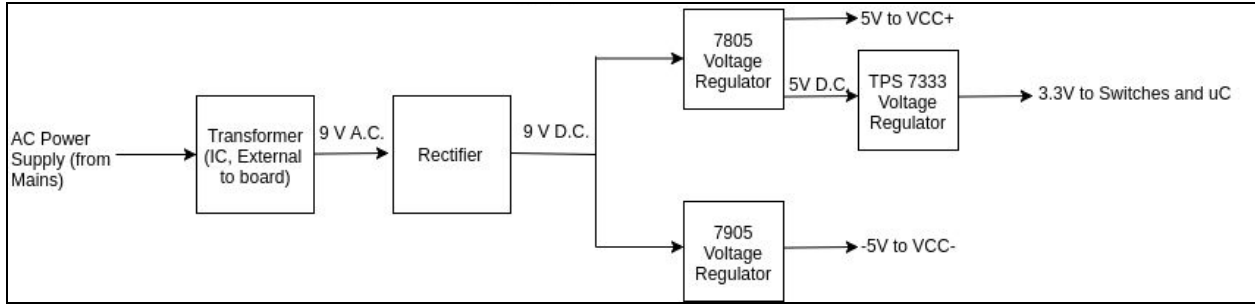


Figure: Block diagram of power management subsystem

ii) Programmable Microcontroller

The analog computer grid contains a large number (110) of analog switches to control the coefficients of the differential equation to be solved. These switches can be configured using a microcontroller which uses SPI communication to communicate with switch and set its state to on or off.

iii) Analog Computer Grid

The analog computer has three main stages - inverter, adder and integrator. The inverter inverts each of the signals x_1 to x_5 . The matrix A involves taking linear combinations of the signals x_1 to x_5 and then adding it with $u(t)$ to generate $Ax + u$. Finally these signals are integrated and fed back to x_1 to x_5 . This implements the system $\dot{x} = Ax + u$.

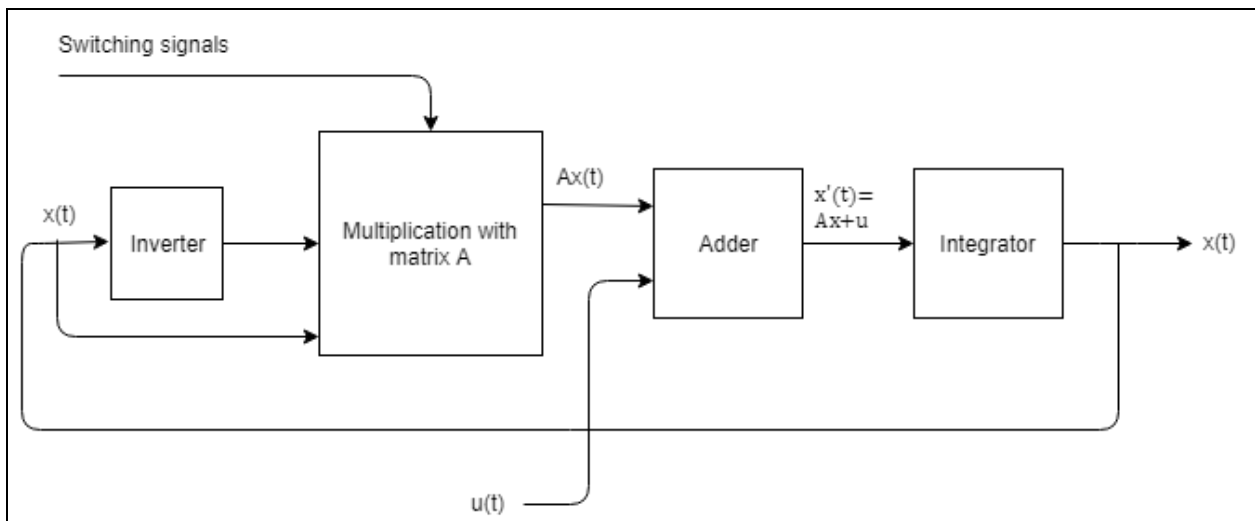


Figure: Overview of the analog computer

3 : Project Implementation

3.1 Power Management Circuit

The input to the power supply circuit is AC voltage of 9V from a transformer. A diode bridge rectifier has been used to convert this to DC. ICs 7805 and 7905 have been used to derive +5V and -5V DC voltages. The op-amps and analog switches are powered by +5V and -5V. A low-dropout (LDO) voltage regulator TPS7333 is used to obtain a 3.3V supply for the microcontroller. This circuit lies on the main board of our design.

Decoupling capacitors of 0.1uF are placed at power supply pins of every IC. Larger decoupling capacitors of 10uF and 1000uF are placed around the 7805 and 7905 ICs. The operating range of 7805 and 7905 requires their input voltage to be at least 8V. Reverse-biased diodes are added to prevent any back-voltage.

A low ESR (1Ω) solid tantalum capacitor must be placed between the output of the TPS7333 and ground. In the absence of this, the output gives a higher value (3.6V) which could damage the microcontroller.

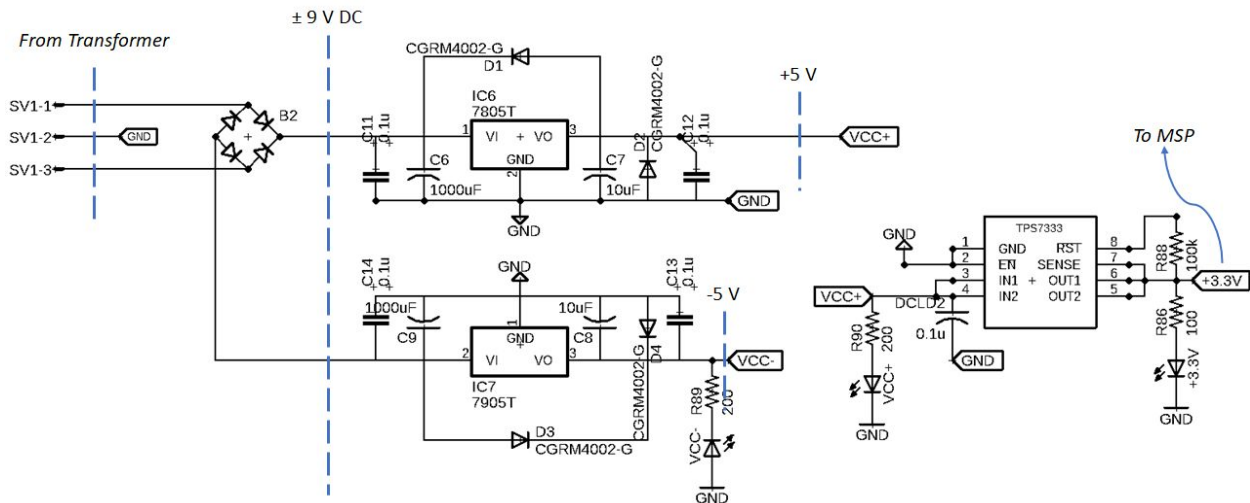


Figure: Design of the power management subsystem

3.2 Microcontroller - MSP430 and MAX395 Switches

DPACv1.0 allows the user to keep 4 different signed values of coefficients ($\pm 0,1,2,3$) of each element of the matrix. This has been implemented by using two resistors in parallel with each other, switched using an SPST switch in series with both the resistors

to control there parallel or separate connections. The switches used are MAX395 ICs, each IC having 8 analog SPST switches. The switches are programmed by sending 8 bits (1 Byte) using SPI. These switches have a maximum on-state resistance of 100Ω. The MAX395 switches are controlled in cascade by SPI communication using MSP430F168 microcontroller, using a process called daisy chaining (explained later).

MAX395 can be considered as a 8 bit shift register. The special thing about these digitally controlled switches is that they can be daisy chained with each other. The input data appearing at the *DIN* port of the register is clocked in at rising edge of *SCLK* and clocked out at the *DOUT* at the falling edge of *SCLK*. The data at *DOUT* is simply *DIN* delayed by 8 clock cycles. As can be seen in the figure below, the *DOUT* pin of a switch may be connected to the *DIN* of another to transfer the register data. The daisy chained switches and the microcontroller communicate with each other using SPI communication.

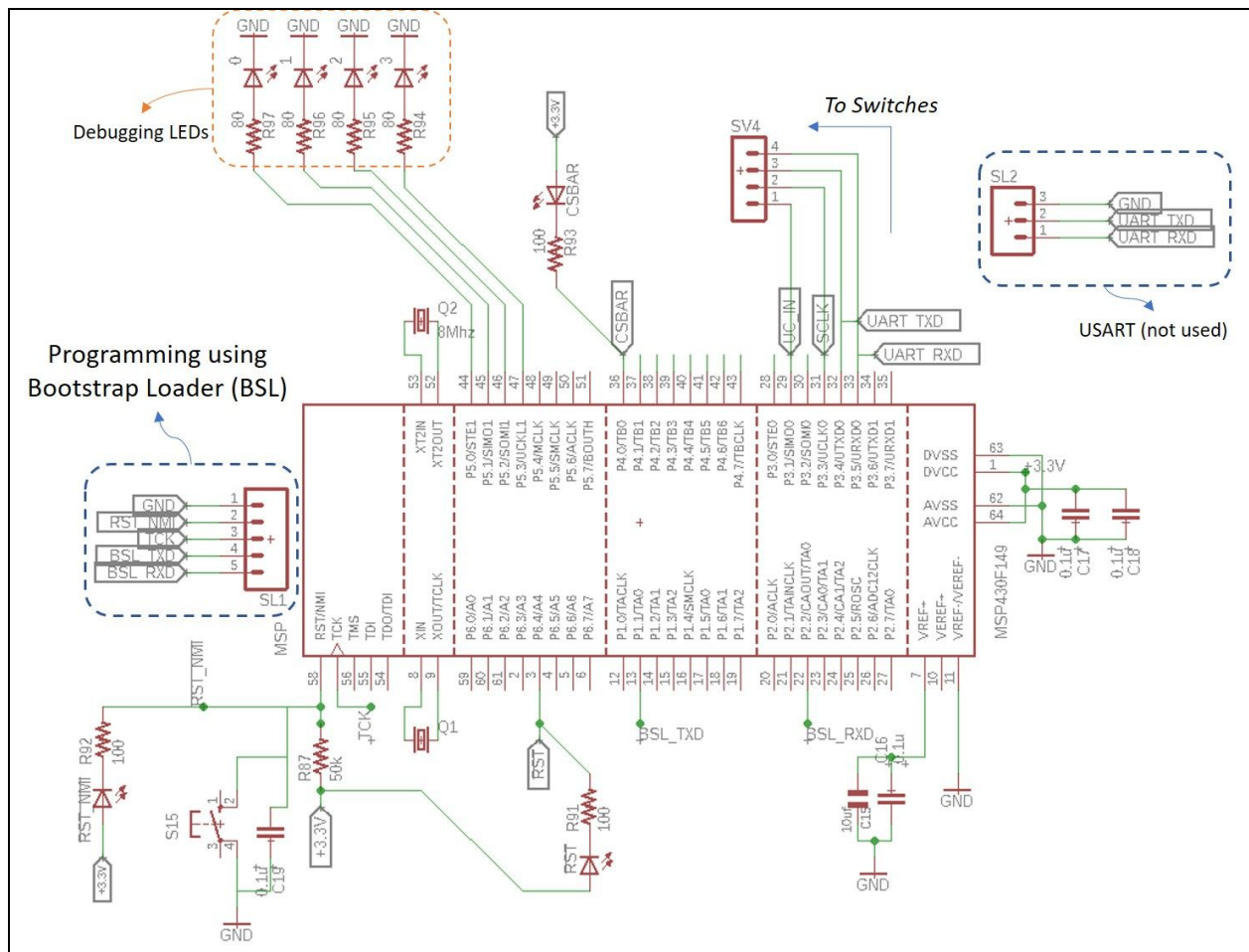


Figure: Setting up the microcontroller

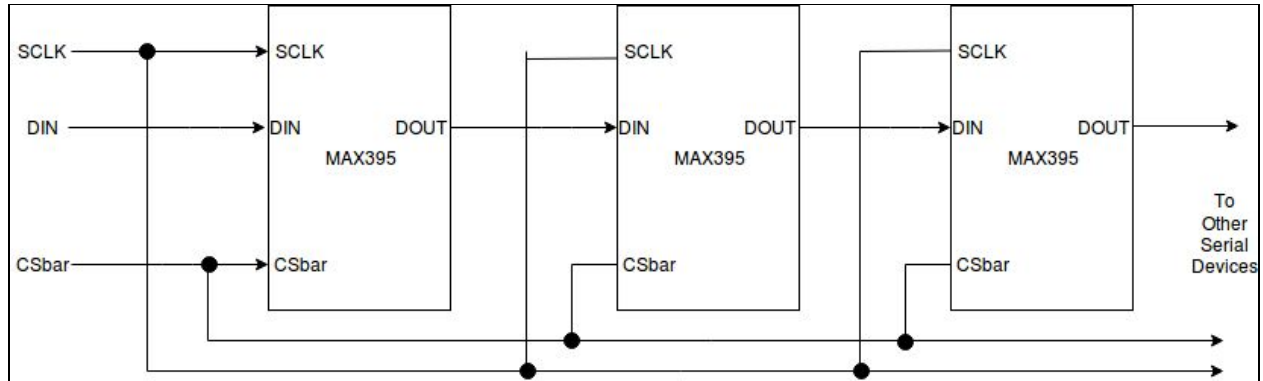


Figure: Daisy chaining of the switches for SPI communication

3.3 Using MSP430F168 for SPI Communication with Slave

MSP has 2 dedicated USART modules which support serial data communication (both synchronous and asynchronous).

1. We use the USART 0 module on *PORT3* for SPI communication.
2. P3.1 is the *SIMO* pin and is connected to the *DIN* of the max395 switches.
3. P3.3 is the *SCLK* and is connected to the *SCLK* of switch
4. The registers of switch clock in data when \overline{CS} is low. We control \overline{CS} using P4.0 of MSP as an output pin. The states of the switches are set according to the contents of the shift register when \overline{CS} goes high.

3.4 Programming the MSP over USB

The Texas Instruments' MSP430 (see figure above) can be programmed using either JTAG or Bootstrap Loader (BSL). We use BSL for programming, using the FT232. The FTDI chip is an interface between the USB port on the PC and the serial port on MSP. The FT232 gives 5 signals to communicate with MSP :-

- BSL_{TXD} Transmits the data to MSP
- BSL_{RXD} Receives the data from MSP
- \overline{RST}_{NMI} For initializing the MSP
- *DTR* For initializing the MSP
- *GND* Ground terminal

The MSP enters bootstrap loader mode when it receives a low pulse on \overline{RST}_{NMI} followed by a low pulse on *DTR*. The PC then sends 0x80 on *RxD* and MSP replies with 0x90 on *TxD*. After this, the program is transferred using *TxD* and *RxD* pins.

3.5 Active devices

As mentioned in the previous section, the three main stages of the analog grid are the inverters, adders and integrators. All three of these are implemented using op-amps and passive elements. For our 5x5 system, the first section has 5 analog inverters to invert the signals x_1 to x_5 . This is followed by 25 switches to choose either x or $-x$ depending on the sign of each coefficients. This is followed by the adder stage. We have 5 adders here. The first adder adds x_1 to x_5 and u_1 scaled by different coefficients, and so on. Thus we can implement even a fully dense matrix. The coefficients are chosen by varying the input resistors of each signal. We have 4 possible values (0,1,2,3) for each coefficient by having 2 resistors, their parallel combination and an open circuit.

The output of the adders goes to the next section with 5 integrators. The gain of the integrator depends on the frequency of the signals used. For instance when a signal $A\sin(2\pi ft)$ is integrated, we get $A\frac{1}{RC}2\pi f\cos(2\pi ft)$. In order to have a reasonable gain for a large range of frequencies, we have made the possibility of changing the resistance at the input of the integrator. We have two values of R which give unity gain at frequencies 330Hz and 3300Hz. However it will give a reasonable gain (close to 1) for frequencies in the order of these frequencies.

Finally, the equation simulated by this system can be represented as:

$$\dot{x} = 2\pi f_0(Ax + u) \text{ where } 2\pi f_0 = \frac{1}{RC} \text{ and } A \text{ can have values in } \{\pm 0, 1, 2, 3\}.$$

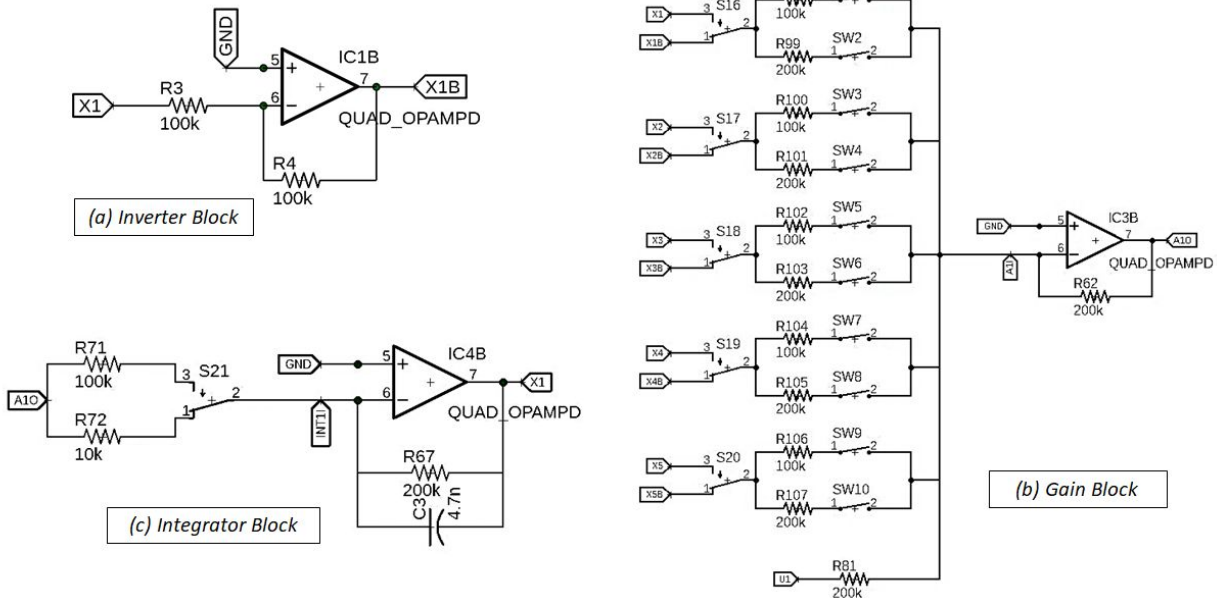


Figure: (clockwise) The inverter block (a) enables signed entries in the system matrix A . The gain block (b) is capable of providing gains within a subset of discrete values. The switches can be programmed to achieve the desired system matrix (signed). The integrator block (c) is responsible for closing the loop on the feedback system, allowing the system to stabilize at its solution.

4 : Performance Evaluation

4.1 First prototype: DPAC- β

Given the complexity of the desired system and the large lead times involved in the planning, layout, design and fabrication of the main board, a preliminary prototype - hereby referred to as the DPAC- β , was designed. The specifications of DPAC- β are as follows

- Can solve linear dynamical systems of upto second order
- The system matrix A is programmable
- Supports a wide range of frequencies, from 350 to 3500 Hz

In this section, we describe the design philosophy and layout of our first prototype for a linear dynamical system solver.

4.1.1 Design

The motivation behind creating the DPAC- β was to understand the behaviour and functioning of a switched dynamical system on a printed circuit board. The prototype is powered by stable DC voltage sources, and the power management circuit is omitted. The system consists of two state variables x_1, x_2 and upto two forcing functions u_1, u_2 . The layout is kept clean, symmetric and easy to interpret and debug.

TL32PO switches are used to emulate the MAX395 analog switches, and can be used to *program* the system. The switches are used to set the entries of the system function A (signed entries $\in \pm \{1, 1.33\}$), as well as the frequency of operation. The figure below shows the schematic of the analog grid in DPAC- β , with the various blocks marked annotated.

The board layout of DPAC- β , which comprises of a dual-layer PCB, further outlines the modular operation and the block-wise design philosophy, making it convenient to debug and localize faults. The PCB was fabricated at the PCB Printing Laboratory within the Department of Electrical Engineering at IIT Bombay. The op-amps used were operated at a V_{cc} of 2V.

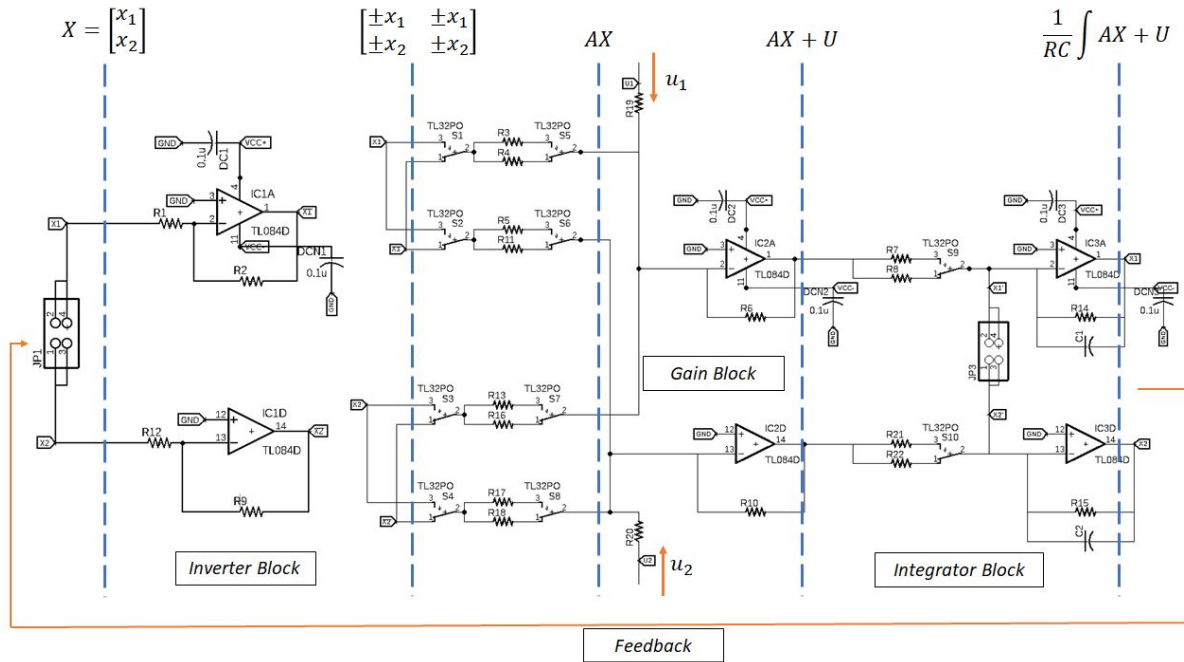


Figure: Annotated circuit layout of DPAC- β

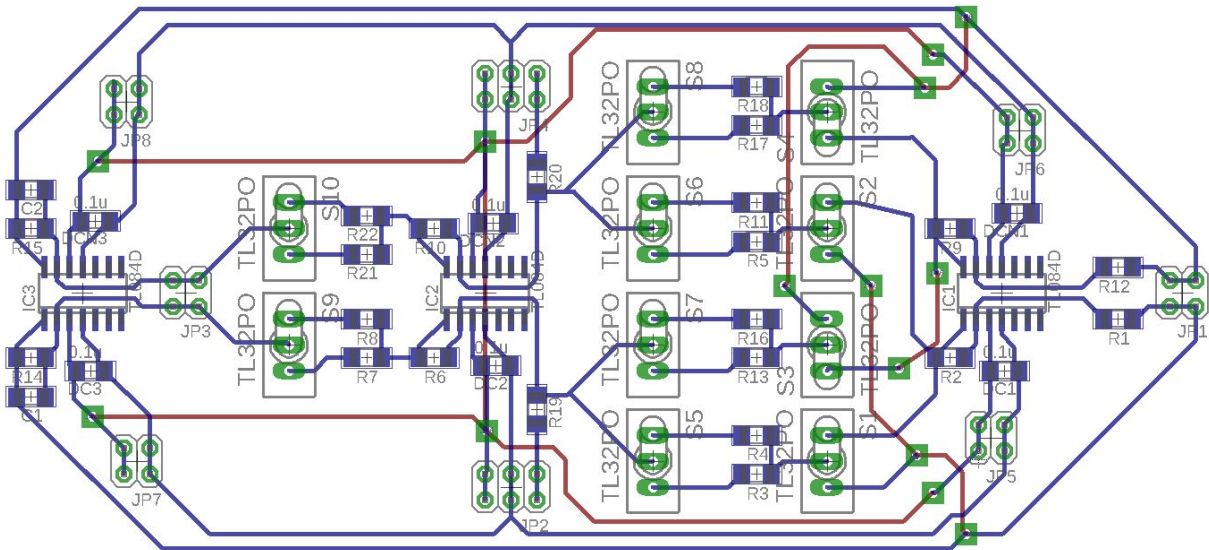


Figure: Board layout of DPAC- β

4.1.2 Experiments and Results

The DPAC- β was extensively tested to model the behaviour of switched dynamical systems in hardware via a routine of cleverly designed experiments. These experiments test each component of the analog grid in DPAC- β , eventually to be used in DPACv1.0, and give valuable insights into the working of an analog linear dynamical system solver.

Here, we present the results and analysis of a set of 4 experiments carried out on the DPAC- β .

In experiments 1 & 2 below, we test each channel of the system independently, i.e., forcing the other state variable to zero. Experiments 3 & 4 demonstrate a full 2×2 system matrix, which is manually configured using the TL32PO switches. The system matrix was designed such that the eigenvalues lie on the open left-half plane, i.e., the system has a stable response. The observed steady-state responses are compared with the *expected* (ideal) steady state responses, which are simulated using Mathematica 11.

1. Single channel setup 1

○ Setup:

- $\dot{X}_1 = -2\pi f_0 A X_1 + u$
- $u : 0.174 \sin(2\pi f t)$
- $A = 1, f_0 = 3386 \text{ Hz}$

○ Steady state solution:

- Expected: $X_2 = 0.1216 \sin(\omega t - 45^\circ)$
- Observed: $\hat{X}_2 = 0.122 \sin(\omega t - 43.8^\circ)$

2. Full matrix setup 1

○ Setup:

- $\dot{X} = 2\pi f_0 (A X + u)$
- A :

$$\begin{bmatrix} -1 & -1.33 \\ 1 & -1 \end{bmatrix}$$

-
- $u : 0.086 [\cos(2\pi f t) \sin(2\pi f t)]^T$
- $f_0 = 338$

○ Steady state solution:

- Expected (In phasor representation):
 - $X = [1.055 \angle 10.39^\circ \quad 0.93 \angle -82.9^\circ]^T$
 - This is relative to the input. The ratio is $1.15 \angle 93.29^\circ$
- Observed:
 - $\hat{X} = [0.244 \angle 0^\circ \quad 0.208 \angle 102^\circ]^T$
 - The ratio of the output is hence $1.13 \angle 102^\circ$

It is important to note that comparing the absolute values of the internal states with the simulated response doesn't have much meaning (in experiments 3 & 4), but the ratio of the phasors must be preserved.

4.1.3 Observations & Inference

Each of the four experiments ran successfully, with their outputs satisfactorily close to the ideal/simulated responses. The following general observations were made:

- A critical requirement of the system is that the system matrix A must be realizable with the available coefficient choices. Using the set available in DPAC- β , the number of experimental setups was limited by the number of stable systems that may be designed with the available coefficients.
- It is important that the forced input is narrow-band, or pure sinusoid, as the active integrator has a frequency-dependent gain. If the frequencies are not of the order of f_0 , the amplification of the signal may be too high (leading to saturation) or too low (will be affected by noise).
- The transients observed decay *very* fast. The transient time is indeed a function of the system parameters. Specifically the transient time constant is of the order of f_0 . This observation gives keen insight into the choice of system parameters for DPACv1.0, so as to allow an observable transient, as per the system requirements.
- The PCB tracks are unshielded and pick up stray signals from the surroundings, affecting the performance of the solver. Since the forcing functions have small magnitudes, this interference hampers the performance of the system.

4.2 DPACv1.0

4.2.1 Features and Specifications

The final board DPACv1.0 comprises of a full-fledged implementation of a digitally programmable linear dynamical system solver, with capability to solve systems upto order 5. It consists of an onboard microcontroller - the Texas Instruments MSP430/F168 - which is programmed via USB. The board also supports communication via USART. The board features its own power management circuitry, and can be plugged into a wall-socket AC source when coupled with a 9V step-down transformer. The board was fabricated at PCB Power Market, Ahmedabad. It intends to solve 5x5 systems (while the DPAC- β solved 2x2 systems).

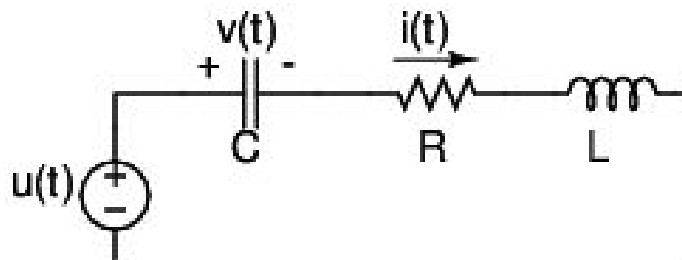
4.2.2 Testing and Evaluation of the Final Board

We tested the board in multiple stages. A systematic way of testing helps us in moving from smaller to larger subsystems. This allows us to efficiently isolate the cause of any error on the board.

- 1) First, we checked the power supply subsystem. If this goes wrong, it could seriously affect the ICs. Thus we checked the power subsystem even before soldering any of the ICs. We checked that every pin and every pad has received the right power supply voltages.
- 2) Second stage was testing the MSP programmer. Unfortunately in this stage, the programmer gives a “Password not recognized” error which we were unable to resolve. Thus we modified our plan to now use an off-board Pt-51 microcontroller to program the switches.
- 3) In the third stage, we tested the control of the switches. At first, we gave a single byte on SPI to test a single MAX 395 IC (8 switches) being programmed correctly. Then we sent 2 bytes to program 2 ICs by daisy chaining. Once this was successful, we sent 14 bytes on SPI to program all the switches.
- 4) After all these components were tested, we move on to test a few representative systems on the DPAC. We begin by evaluating this board on 1x1, then 2x2 systems, and use the same test cases that we used for DPAC- β . After testing that each of the channels X_1 to X_5 are working, we proceed to test some higher-order systems.
- 5) Lastly, we simulated a few real-world dynamical systems on the DPAC, and paid keen attention to the transient responses. The results of these experiments will be outlined below.

4.2.3 Experiments and Results

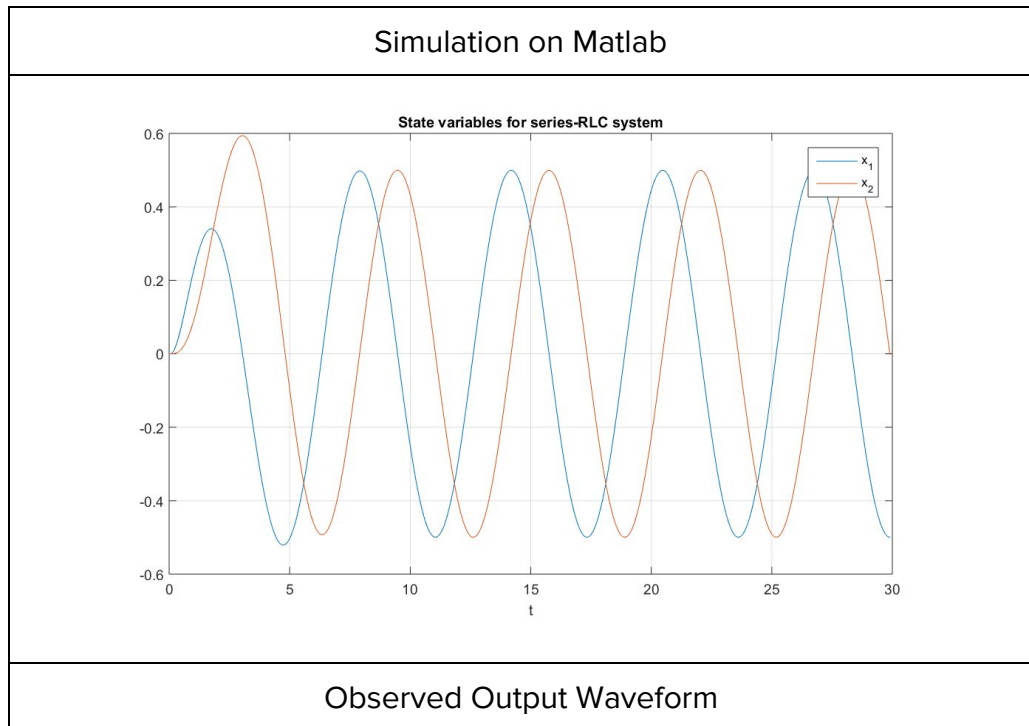
1. Simulation of sinusoidal excitation of a series R-L-C circuit

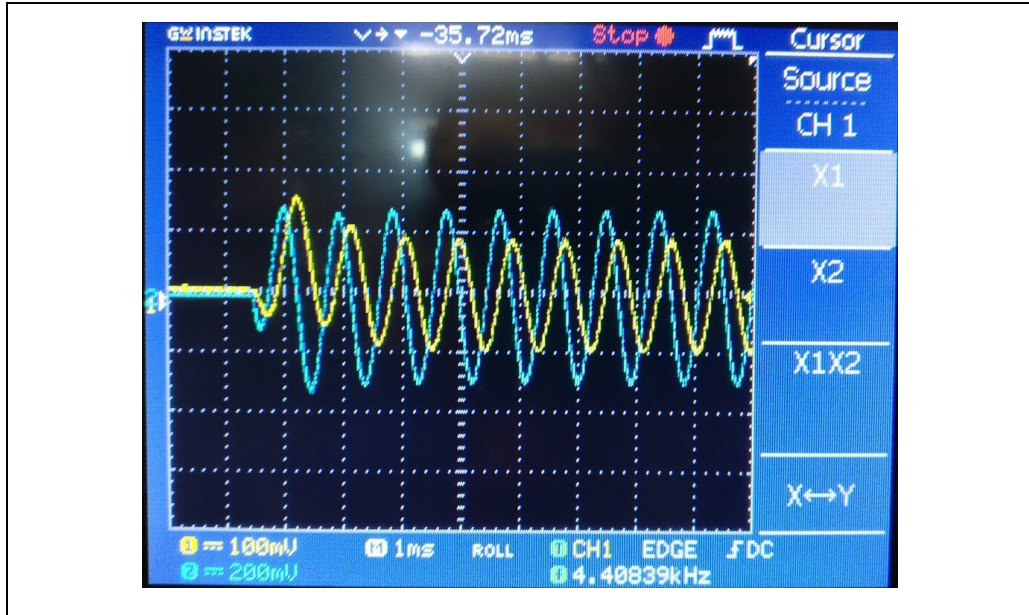


- Setup:

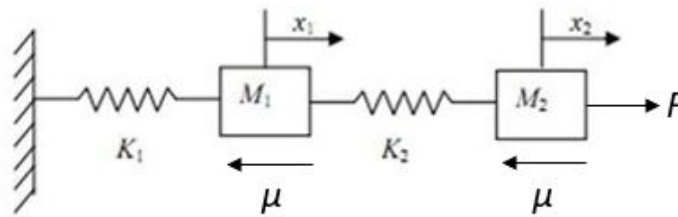
- $u = Ri + L \frac{di}{dt}, i = C \frac{dv}{dt}$
- $R = 2, L = C = 1$

- $X = [i \ v]^T$
- $\dot{X} = 2\pi f_0(AX + u)$
- **A :**
$$\begin{bmatrix} -2 & -1 \\ 1 & 0 \end{bmatrix}$$
- $u : [0.5\sin(2\pi ft) \ 0]^T$
- $f_0 = 338$
- Solution:
 - The transients may not match exactly because the initial phase of the forcing function obtained from the AFG is random.





2. Simulation of a spring-mass system with friction



○ Setup:

■ $m_1 \frac{dv_1}{dt} = -K_1 x_1 - \mu_1 v_1$, $m_2 \frac{dv_2}{dt} = F - K_2(x_2 - x_1) - \mu_2 v_2$

■ $v_1 = \frac{dx_1}{dt}$, $v_2 = \frac{dx_2}{dt}$

■ $K_1 = K_2 = \mu_1 = \mu_2 = m_1 = m_2 = 1$

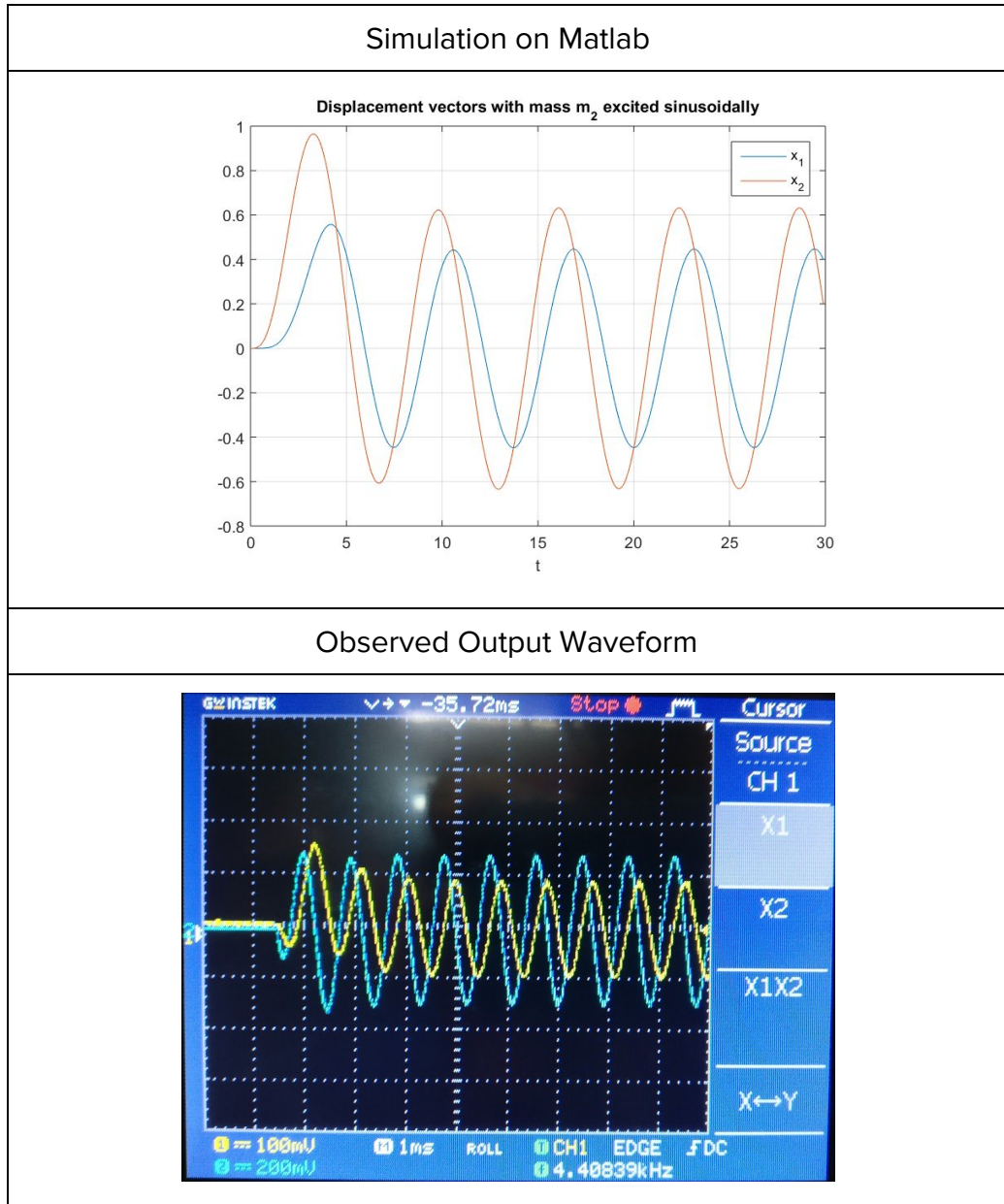
■ $X = [x_1 \ x_2 \ v_1 \ v_2]^T$

■ $\dot{X} = 2\pi f_0 (AX + u)$

■ A :

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -2 & 1 & -1 & 0 \\ 1 & -1 & 0 & -1 \end{bmatrix}$$

- $u : F = [0 \ 0 \ 0 \ 0.28\sin(2\pi ft)]^T$
- $f_0 = 338$
- **Solution:**
 - This experiment illustrates the capability of the DPAC to simulate higher order systems.
 - The transients may not match exactly because the initial phase of the forcing function obtained from the AFG is random.



3. Simulation of step response of a second order transfer function

○ Setup:

- $G(s) = \frac{1}{s^2 + 2\zeta\omega_n s + \omega_n^2}$
- $X = [x \ \dot{x}]^T$
- $\dot{X} = 2\pi f_0 (AX + u)$
- $u : [0 \ u(t)]^T$ (Step function was realised using a 4Hz pulse with 2.5% duty cycle)
- $f_0 = 338$
- Case 1: $\omega_n^2 = 3$, $\zeta\omega_n = 1$ (Lesser damping)

A :

$$\begin{bmatrix} 0 & 1 \\ -3 & -1 \end{bmatrix}$$

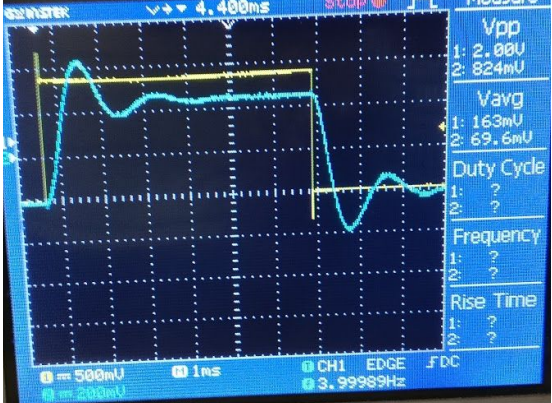
- Case 2 : $\omega_n^2 = 3$, $\zeta\omega_n = 2$ (More damping)

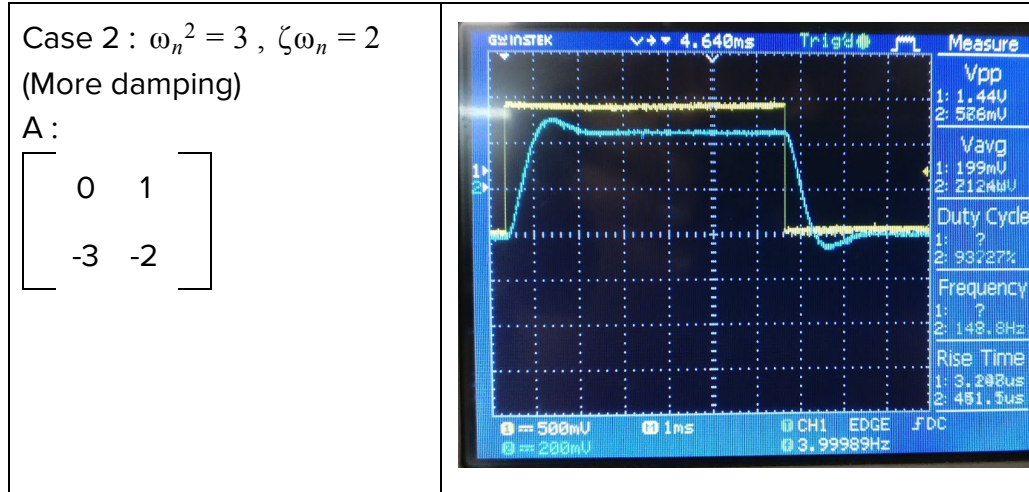
A :

$$\begin{bmatrix} 0 & 1 \\ -3 & -2 \end{bmatrix}$$

○ Solution:

- This experiment was used to illustrate the programmability of the DPAC. We can easily change the parameters of the system matrix to change the damping ratio of the step response.

Test Case	Observed Output Waveform
<p>Case 1: $\omega_n^2 = 3$, $\zeta\omega_n = 1$ (Lesser damping)</p> <p>A :</p> $\begin{bmatrix} 0 & 1 \\ -3 & -1 \end{bmatrix}$	



4. Summary of the experiments:
 - a. We have successfully tested system of upto the 5th order on the DPAC.
 - b. We have found that the transient response is also very accurate at any frequency of operation (in the range of the DPAC).
 - c. The only limitation of the DPACv1.0 is that the coefficients are chosen from a small discrete set of values.

5: Conclusion & Future Work

This project may be taken as a proof-of-concept for a digitally programmable analog computer. As follow ups to this project , following tasks can be done :-

- 1) Rectify the programing error of MSP microcontroller, so that we can have an on-board microcontroller.
- 2) Extend the design to allow modeling of nonlinear systems by using the onboard microcontroller, with DAC & ADC, as multipliers etc.
- 3) Explore an alternative design which can allow us to program a wider range of coefficients. (Example - use of digital potentiometers). We did not used digital potentiometers in DPACv1.0 because they come with poor tolerances.
- 4) We have not taken into account equations which may have non trivial initial conditions. This can be incorporated by implementing a design which has options for proper charging and discharging of capacitors on board.

APPENDIX

A: MSP430 Issues and Pt51

We initially had planned to use MSP430 on board along with a FTDI board to program all the MAX395 switches. MSP430 has several advantages over other microcontrollers like Pt51 and arduino foremost of which is that it has multiple UART communication modules. MSP430 has 8 ADC pins which can be used for implementing on board multiplication and later for solving higher degree differential equations.

We use FTDI board to program the MSP via BSL interface. Hex files can be directly uploaded on MSP via Linux Terminal. One just needs to install msp430 gcc compiler for this purpose (Linux Terminal :- `sudo apt-get install gcc-msp430`).

When connecting with the FTDI for the first time, you get a error related to driver file for USB port 0. It can be easily resolved by googling the error and updating the related file. Unfortunately we could not resolve some issues in connecting with MSP430 which persistently showed “password not recognized” error. It was due to MSP not responding back to the communication signals sent by FTDI board. The RST/NMI pin as well as the TCK pin did get the required pulses and the BSL_TXD sends the first bit for checking communication but the MSP doesn't respond back. Reasons for this unexpected behaviour could not be figured out.

The reader is highly encouraged to try working with MSP and figure out the error. After discussion with our faculty guide and some seniors from the lab, we speculate that the error is because of an improper power-on reset of the MSP. One attempt to resolve the issue can be to connect the reset output of the TPS7333 LDO voltage converter to the reset pin of the MSP.

The current version i.e. DPAC 1.0 uses a Pt51 microcontroller board (a 8051 based in-house production of IIT Bombay). A proper documentation on its usage is easily available in Wadhvani Electronics Laboratory (WEL, IITB).

The Pt-51 is easily programmed in C language using an assembler like Keil. The Hex file generated is loaded on the Pt-51 using Atmel Flip. It is done via a direct USB connection.

B: Setting up DPACv1.0 - A User's Guide

Powering up the DPACv1.0 is very simple. Follow this guide to simulate your own linear dynamical system on the DPAC!

Power Supply

Arrange for a DC supply of +9V and -9V. You may need to use 2 separate power supplies for the same. Connect +9V, GND and -9V to red, black and green header cap respectively on the DPAC. Alternatively, you can also connect any AC power source with a peak voltage of around 9V, with connections made accordingly.

Microcontroller Connections - Pt51 Development Board

- 1) SCLK on Pt51 to SCLK header on DPAC
- 2) MOSI of Pt51 to UC_IN header on DPAC
- 3) P3.0 of Pt51 to CS_bar (Any other GPIO pin maybe used in place of P3.0 along with appropriate changes made in the code)
- 4) GND to GND
- 5) RESET_BAR header to +3.3V on the DPAC board

Configuring the System Matrix and Forcing Functions

- 1) We have made a simple Python script which can be used to get the switch configurations that need to be programmed on the DPAC, given the system matrix.
- 2) These values are put into the C code for the Pt-51 microcontroller. The code is compiled and built using Keil.
- 3) Finally the *.hex* file generated is loaded on the Pt-51 using Atmel Flip.
- 4) A set of 6 pin headers towards the bottom of the board provide pins to connect the 5 forcing functions u_1 to u_5 and GND.
- 5) A set of 5 pairs of pin headers towards the bottom left provide pins to observe the outputs x_1 to x_5 .