A MAJOR PROJECT REPORT

ON

OPTIMIZATION OF CONICAL TANK SYSTEM THROUGH MACHINE LEARNING BASED CONTROLLER

Submitted in partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

In

ELECTRONICS AND INSTRUMENTATION ENGINEERING

Submitted by

Kausthuba Vanam	**17B81A1019**
M. Rukma Reddy	**17B81A1036**
Ch. Vijendar Reddy	**17B81A1058**

Department of Electronics and Instrumentation Engineering

CVR COLLEGE OF ENGINEERING

(An UGC Autonomous Institution with NAAC 'A' Grade Accredited by NBA)

Approved by AICTE & Affiliated to JNTU Hyderabad

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M), RR District, Telangana, India, PIN- 501 510

2017-2021

A MAJOR PROJECT REPORT

ON

OPTIMIZATION OF CONICAL TANK SYSTEM THROUGH MACHINE LEARNING BASED CONTROLLER

Submitted in partial fulfillment for the award of degree

Of

BACHELOR OF TECHNOLOGY

In

ELECTRONICS AND INSTRUMENTATION ENGINEERING

Submitted by

Kausthuba Vanam	**17B81A1019**
M. Rukma Reddy	**17B81A1036**
Ch. Vijendar Reddy	**17B81A1058**

Under the guidance of

Dr. Bellamkonda Saidulu

Associate Professor

Department of Electronics and Instrumentation Engineering

CVR COLLEGE OF ENGINEERING

(An UGC Autonomous Institution with NAAC 'A' Grade Accredited by NBA)

Approved by AICTE & Affiliated to JNTU Hyderabad

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M), RR District, Telangana, India, PIN- 501 510

2017-2021

CVR COLLEGE OF ENGINEERING

(An UGC Autonomous Institution with NAAC 'A' Grade Accredited by AICTE)

Department of Electronics and Instrumentation Engineering

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M), R.R. District, Telangana 501510

CERTIFICATE

This is to certify that the dissertation work entitled **"OPTIMIZATION OF CONICAL TANK SYSTEM USING MACHINE LEARNING BASED CONTROLLER**" is the work completed and submitted by

Kausthuba Vanam	**17B81A1019**
M. Rukma Reddy	**17B81A1036**
Ch. Vijendar Reddy	**17B81A1058**

This report is submitted in the partial fulfillment of the academic requirements for the award of the degree of '**BACHELOR OF TECHNOLOGY**' in '**Electronics and Instrumentation Engineering**' from **CVR College of Engineering** affiliated to Jawaharlal Nehru Technological University, Hyderabad is a bonafide project work carried out by them during the academic year 2020-2021.

Dr. Bellamkonda Saidulu	**Dr. S.Harivardhagini**
PROJECT GUIDE	**HEAD OF THE DEPARTMENT**

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mentioning of the people whose constant guidance and encouragement made it possible.

We express our utmost gratitude to **Dr. K.S.Nayanathara**, Principal of our college, for constant encouragement and motivation throughout the academic.

We derive great pleasure in expressing our sincere gratitude to our **Head of the Department Dr.S.Harivardhagini** for her timely suggestions which helped us to complete this work successfully

We express our sincere thanks to our internal guide **Dr. Bellamkonda Saidulu, Associate professor,** Department of EIE, for giving us support, kind attention and valuable guidance throughout the project.

ABSTRACT

Conical tank is a non-linear control system, whose objective is to control the level of the liquid in the tank. Controlling the level of a liquid in such tanks is a challenging task because of its non-linearity and the continuously changing area of cross section. Reinforcement learning is a part of machine learning, where an agent in an environment learns to take the desired actions on its own based on the reinforcement inputs which are usually referred to as rewards or punishments. In this process output valve is made constant and inflow of liquid is controlled by the pneumatic control valve. The machine learning based controller must give correct input to control valve, so that the inflow of liquid is such that the level is controlled at the desired set point. The system is based on the principle of Deep Deterministic Policy Gradient (DDPG) which is represented by a reinforcement learning toolkit with Actor-Critic learning algorithm inbuilt in the MATLAB Simulink environment to control the process. This controller provides an edge over PID, fuzzy, and other neural network based controllers, by eliminating the need for linearizing non-linear characteristics, tuning PID parameters and designing transfer functions. The designed intelligent controller prevents the system from damaging due to flow disturbances, rejects unknown disturbances and settles liquid level in a shorter time

TABLE OF CONTENTS **PAGE NO**

LIST OF FIGURES **PAGE NO**

LIST OF TABLE **PAGE NO**

CHAPTER 1

INTRODUCTION

Conical tank is a classical non-linear control problem whose objective is to control the level of the liquid in the tank. Conical tanks not only assure better mixing and stirring of the ingredients but also the conical shape ensures efficiency in cleaning and drainage of solid wastes and thick slurries. Controlling the level of a liquid in such tanks is a challenging task because of its nonlinearity and the continuously changing area of cross section [1-12]. Conical tanks are used in many process industries like waste water treatment industries, food industries, chemical process plants, petroleum refineries, and pharmaceutical industries. The liquid level in such industries must be precisely controlled for maintaining the chemical equilibrium of various processes.

The common need for accurate and efficient control of today's industrial applications is driving the system identification field to face the constant challenges of providing better models of physical phenomena. Systems encountered in practice are often nonlinear or have time varying nature. So, it is difficult to identify accurate models of a nonlinear system. Approximate linear models are used in most of the industrial controllers, which may lead to lower the control performance. Hence, it is important to develop a simple and practical method for nonlinear process modeling and identification, and use that conical tank system in various industries for control of non linear processes.

There are many existing control methods for controlling a non-linear conical tank process. In addition to PID and fuzzy techniques various neural networks and adaptive control techniques have also been suggested. Although several methods have been developed, in most of them non-linear parameters are linearized before the controller is designed. In case of PID and fuzzy controllers, the major drawback is the large number of parameters to be tuned. Without any assumptions, linearization, and modeling of system parameters, an efficient and robust controller is designed. Reinforcement learning is a Machine learning technique being extensively used in robotic applications is advantageous as it is less complex, and doesn't require the modeling of the system.

1.1 OBJECTIVE

The main objective of our project is to control the level of the liquid inside the tank. This is achieved by increasing the inflow of liquid from the reservoir or by changing the output valve. In this experimental process, output valve is made constant, and is unaltered during the experimentation. Inflow of liquid is controlled by the pneumatic control valve. The Reinforcement learning toolkit controller in Matlab Simulink follows trial and error method and should be able to give correct input to the control valve after the model is trained, so that the inflow of liquid is such that the level is controlled at the desired set point.

1.2 LITERATURE SURVEY

Each and every industry phase the flow control and level control problem. So that conical tank level process is used. Canonical tank level process is used. Canonical tank is the highly non-linear system. Because of this, reason sometimes output may be affected. Due to its shape, the conical tank is lead to non-linearity control of conical tank is the challenging problem so many researchers have been carried out in the level control of conical process. Several controllers like PID (Proportional-Integral-Derivative) controllers, hybrid fuzzy logic controllers, neural network based controllers, and a myriad of control techniques have been implemented on the conical tank system, and their results were discussed. Bhuvaneswari, N. S., Uma, G., & Rangaswamy, T. R. [1] proposed a time optimal control for set-point changes, and an adaptive control for process parameter variations using neural network for a non-linear conical tank level process. In Aravind, P., & Kumar, S. G. [3], a non-linear model of the conical tank level control system is presented. In Madhubala et al. [14], the development and tuning of fuzzy controllers for the conical tank system is discussed. Noel and Pandian [15], used a new artificial neural network based reinforcement learning approach for optimal control of a non-linear liquid level system. Thus, many controlling concepts have been studied and implemented on the conical tank system.

Reinforcement learning which is a machine learning based algorithm, is used in robotics and control systems for creating self-learning models. A fast and feasible reinforcement learning algorithm is implemented in Ono, S., Inagaki, Y., Aisu, H., Sugie, H., & Unemi, T. [16]. Boubertakh, H., Tadjine, M., Glorennec, P. Y., & Labiod, S. [17] used reinforcement learning. algorithm to tune PID and fuzzy controllers. As discussed by Yen, G., & Hickey, T.[18], reinforcement learning algorithm is an important tool for precise robotic navigation applications. In Mohajerin, N., Menhaj, M. B., & Doustmohammadi, A. [19], a reinforcement learning based fuzzy logic controller is designed and implemented on a ball and plate system. As discussed

above, there are many existing control methods for controlling a non-linear conical tank process. In addition to PID and fuzzy techniques various neural networks and adaptive control techniques have also been suggested. Although several methods have been developed, in most of them non-linear parameters are linearized before the controller is designed. In case of PID and fuzzy controllers, the major drawback is the large number of parameters to be tuned. Prabhu Ramanathan, Kamal Kant Mangla, Sitanshu Satpathy [20] proposed Q-learning algorithm based on value iteration to design the controller. In order to have a faster convergence, the height of the tank is discretized into small finite equal spaces. The online learning is implemented on a trial basis. Each trial has a learning period, and if the liquid can settle in the desired level, it is considered as a success, otherwise a failure. The next trial now has the knowledge of the previous trial and this process is repeated. After a few trials, the level settles very quickly, and the learning is stopped. The knowledge gained is used to implement a standalone controller.

1.3 CONCLUSION

This chapter provides an idea about Machine learning based controller for optimization of conical tank system. This chapter not only gives a brief overview about the project but also demonstrates the research work on conical tank system by different researchers in controlling the level of liquid. Finally, this chapter conveys the motive and knowledge gained to implement intelligent and standalone controller.

CHAPTER 2

DATA ACQUISITION SYSTEM

2.1 INTRODUCTION TO DAC :

The **Data acquisition** is the process of sampling signals that measure real world physical conditions and converting the resulting samples into digital numeric values that can be manipulated by a computer. Data acquisition systems, abbreviated by the acronyms DAS or DAQ, typically convert analog waveforms into digital values for processing. The components of data acquisition systems include:

- Sensors to convert physical parameters to electrical signals.
- Signal conditioning circuitry, to convert sensor signals into a form that can be converted to digital values.
- Analog -to-digital converters, to convert conditioned sensor signals to digital values

Here we are using ADuC841 microcontroller for data acquisition. The functional block diagram of this microcontroller is given below:

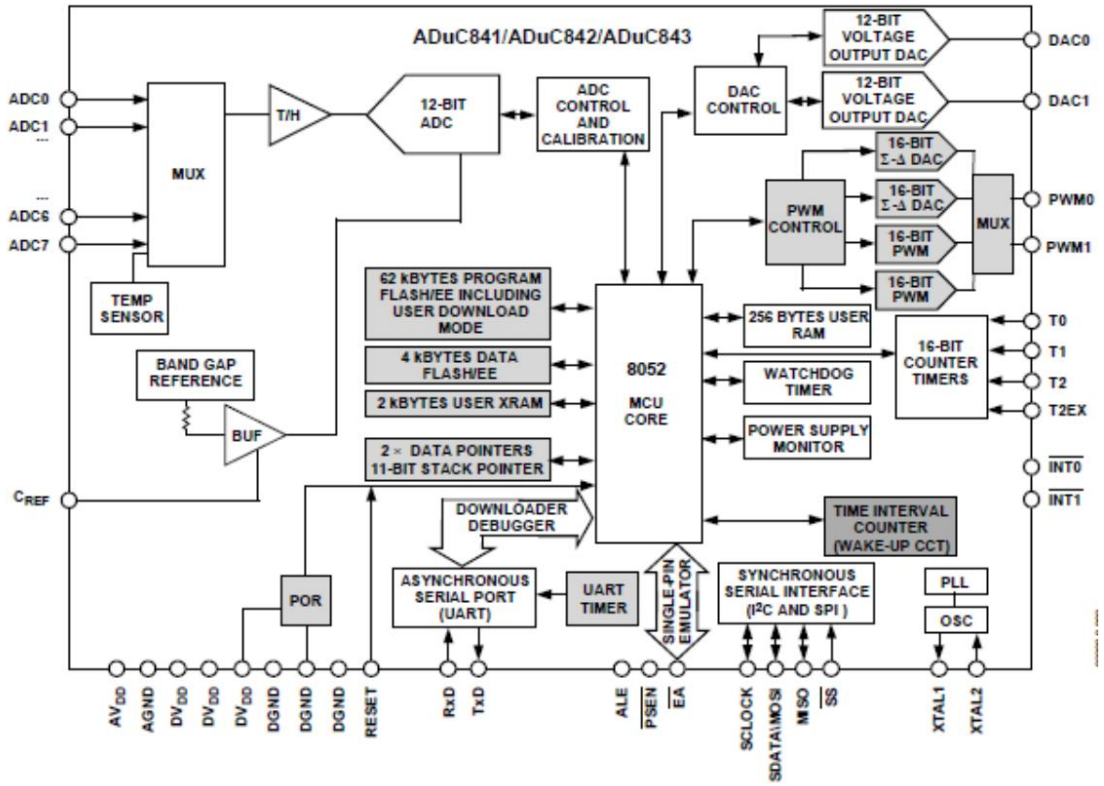


Fig.2.1 Functional Diagram of microcontroller

➢ The Purposes of Data Acquisition

The primary purpose of a data acquisition system is to acquire and store the data. But they are also intended to provide real-time and post-recording visualization and analysis of the data. Furthermore, most data acquisition systems have some analytical and report generation capability built-in.

A recent innovation is the combination of data acquisition and control, where a high quality DAQ system is connected tightly and synchronizes with a real-time control system. Engineers in different applications have a various requirements, of course, but these key capabilities are present in varying proportions:

- Data recording
- Data storing
- Real-time data visualization
- Post-recording data review
- Data analysis using various mathematical and statistical calculations
- Report generation

➢ FEATURES:

- Pin compatible upgrade of ADuC812/ADuC831/ADuC832
- Increased performance
- Single-cycle 20 MIPS 8052 core
- High speed 420 kSPS 12-bit ADC
- Increased memory
- Up to 62 kBytes on-chip Flash/EE program memory
- 4 kBytes on-chip Flash/EE data memory
- In-circuit reprogrammable
- Flash/EE, 100 year retention, 100 kCycle endurance
- 2304 bytes on-chip data RAM
- Smaller package

- 8 mm × 8 mm chip scale package
- 52-lead PQFP—pin-compatible upgrade
- Analog I/O
- 8-channel, 420 kSPS high accuracy, 12-bit ADC
- On-chip, 15 ppm/°C voltage reference
- DMA controller, high speed ADC-to-RAM capture
- Two 12-bit voltage output DACs1
- Dual output PWM Σ-Δ DACs
- On-chip temperature monitor function
- 8052 based core
- 8051 compatible instruction set (20 MHz max)
- High performance single-cycle core
- 32 kHz external crystal, on-chip programmable PLL
- 12 interrupt sources, 2 priority levels
- Dual data pointers, extended 11-bit stack pointer
- On-chip peripherals
- Time interval counter (TIC)
- UART, I2C®, and SPI® Serial I/O
- Watchdog timer (WDT)
- Power supply monitor (PSM)
- Power
- Normal: 4.5 mA @ 3 V (core CLK = 2.098 MHz)
- Development tools
- Low cost, comprehensive development system incorporating nonintrusive single-pinemulation,
- IDE based assembly and C source debugging

➢ APPLICATIONS:

- Optical networking—laser power control
- Base station systems
- Precision instrumentation, smart sensors
- Transient capture systems
- DAS and communications systems

➢ DAS HARDWARE:

DAQ hardware is what usually interfaces between the signal and a PC. It could be in the form of modules that can be connected to the computer's ports (Parallel, serial, USB, etc.) or cards connected to slots (S-100 bus, AppleBus, ISA, MCA, PCI, PCI-E, etc.) in a PC motherboard or in a modular crate (CAMAC, NIM, VME). Sometimes adapters are needed, in which case an external breakout box can be used.

DAQ cards often contain multiple components (multiplexer, ADC, DAC, TTL-IO, high speed timers, RAM). These are accessible via a bus by a microcontroller, which can run small programs. A controller is more flexible than a hard-wired logic, yet cheaper than a CPU so that it is permissible to block it with simple polling loops. For example: Waiting for a trigger, starting the ADC, looking up the time, waiting for the ADC to finish, move value to RAM, switch multiplexer, get TTL input, let DAC proceed with voltage ramp.

DAQ is the process of sampling electrical signals, typically those that measure physical conditions. These systems generally consist of three components, including sensors, signal conditioning circuitry and an analog-to-digital converter. The sensors convert physical parameters into an analog signal. The signal-conditioning circuitry converts signals from the sensors into a form that can be converted into digital values. An analog-to-digital converter then converts the conditioned analog signals to digital values. Stand-alone DAQ systems are typically known specifically as data loggers. Low input impedance data loggers generally have an input impedance on the order of 22 kΩ. The requirement for a data logger with a high input impedance means that it should have an input impedance of at least 100 MΩ, which significantly increases the cost of the unit. Additional features for this type of data logger include an analog-to-digital (A/D) converter with 16-bit successive approximation. It should also have 8 single-ended channels with individual A/D on each channel. Typical ranges for the voltage inputs include ±1V, ±2V, ±5V and ±10V.

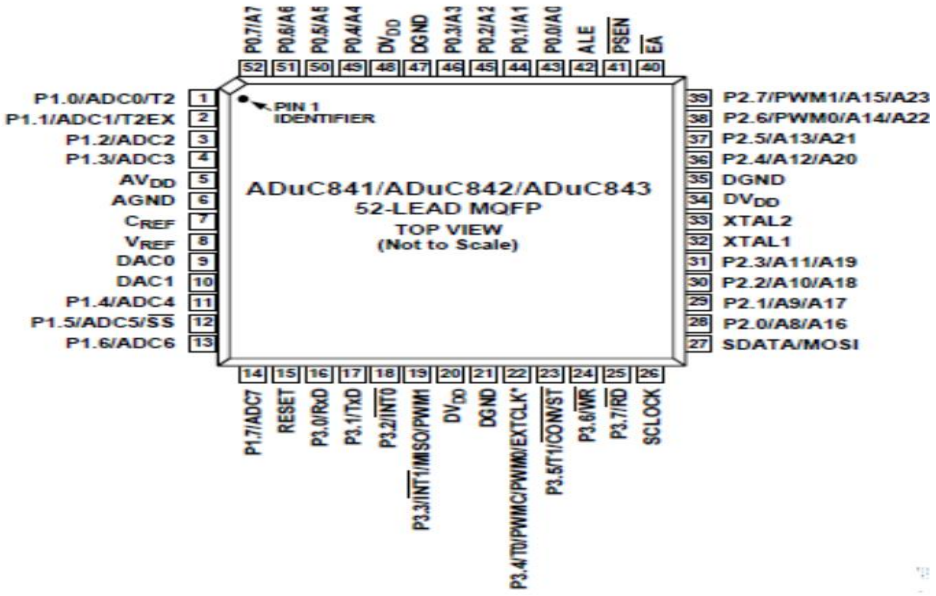


Fig 2.2 Pin diagram of ADuC841 microcontroller

➢ GENERAL DESCRIPTION:

The ADuC841/ADuC842/ADuC8431 are complete smart transducer front ends, that integrates a high performance self-calibrating multichannel ADC, a dual DAC, and an optimized single-cycle 20 MHz 8-bit MCU (8051 instruction set compatible) on a single chip.

The ADuC841 and ADuC842 are identical with the exception of the clock oscillator circuit; the ADuC841 is clocked directly from an external crystal up to 20 MHz whereas the ADuC842 uses a 32 kHz crystal with an on-chip PLL generating a programmable core clock up to 16.78 MHz The ADuC843 is identical to the ADuC842 except that the ADuC843 has no analog DAC outputs. The microcontroller is an optimized 8052 core offering up to 20 MIPS peak performance. Three different memory options are available offering up to 62 Kbytes of non-volatile Flash/EE program memory. Four Kbytes of non-volatile Flash/EE data memory, 256 bytes RAM, and 2 Kbytes of extended RAM are also integrated on-chip.

➢ ADC SPECIFICATIONS

- **Integral Nonlinearity**

  The maximum deviation of any code from a straight line passing through the endpoints of the ADC transfer function. The endpoints of the transfer function are zero scale, a point ½ LSB below the first

code transition, and full scale, a point ½ LSB above the last code transition.

- **Differential Nonlinearity**

  The difference between the measured and the ideal 1 LSB change between any two adjacent codes in the ADC.

- **Offset Error**

  The deviation of the first code transition (0000 . . . 000) to (0000 . . . 001) from the ideal, that is, +½ LSB.

- **Gain Error**

  The deviation of the last code transition from the ideal AIN voltage (Full Scale – ½ LSB) after the offset error has been adjusted out.

- **Signal-to-(Noise + Distortion) Ratio**

  The measured ratio of signal to (noise + distortion) at the output of the ADC. The signal is the rms amplitude of the fundamental. Noise is the rms sum of all nonfundamental signals up to half the sampling frequency (fS/2), excluding dc. The ratio depends on the number of quantization levels in the digitization process; the more levels, the smaller the quantization noise. The theoretical signal-to-(noise + distortion) ratio for an ideal N-bit converter with a sine wave input is given by ,

  Signal-to-(Noise + Distortion) = (6.02*N* + 1.76) *dB*

  Thus for a 12-bit converter, this is 74 dB.

- **Total Harmonic Distortion (THD)**

  The ratio of the rms sum of the harmonics to the fundamental

➢ DAC SPECIFICATIONS

- **Relative Accuracy**

  Relative accuracy or endpoint linearity is a measure of the maximum deviation from a straight line passing through the endpoints of the DAC transfer function. It is measured after adjusting for zero error and full-scale error.

- **Voltage Output Settling Time**

The amount of time it takes for the output to settle to a specified level for a full-scale input change.

- **Digital-to-Analog Glitch Impulse**

  The amount of charge injected into the analog output when the inputs change state. It is specified as the area of the glitch in nV-sec.

2.2 CONCLUSION:

From this we conclude that, DAC is a main device used as an interface between the conical tank instrument and computer for converting analog data into digital data . DAC not only supports one way communication ,it also supports two way communication from Computer to Instrument. Current to voltage is also converted in the DAC system which involves in effective interconnection between two analog and digital devices.

CHAPTER 3
SOFTWARE DESCRIPTION

3.1 MATLAB SOFTWARE

The MATLAB platform is optimized for solving engineering and scientific problems. The matrix-based MATLAB language is the world's most natural way to express computational mathematics. Built-in graphics make it easy to visualize and gain insights from data. A vast library of pre-built toolboxes lets you get started right away with algorithms essential to your domain. The desktop environment invites experimentation, exploration, and discovery. These MATLAB tools and capabilities are all rigorously tested and designed to work together. MATLAB R2021a version is used in optimization of conical tank system, This version contains Reinforcement toolkit which is used as an optimization technique based on actor and critic algorithm.

Millions of engineers and scientists worldwide use MATLAB to analyze and design the systems and products transforming our world. MATLAB is in automobile active safety systems, interplanetary spacecraft, health monitoring devices, smart power grids, and LTE cellular networks. It is used for machine learning, signal processing, image processing, computer vision, communications, computational finance, control design, robotics, and much more. MATLAB helps you take your ideas beyond the desktop. You can run your analyses on larger data sets, and scale up to clusters and clouds. MATLAB code can be integrated with other languages, enabling you to deploy algorithms and applications within web, enterprise, and production systems.

➢ Key Features

- High-level language for scientific and engineering computing
- Desktop environment tuned for iterative exploration, design, and problem-solving
- Graphics for visualizing data and tools for creating custom plots
- Apps for curve fitting, data classification, signal analysis, control system tuning, and many other tasks
- Add-on toolboxes for a wide range of engineering and scientific applications
- Tools for building applications with custom user interfaces
- Interfaces to C/C++, Java®, .NET, Python, SQL, Hadoop, and Microsoft® Excel®
- Royalty-free deployment options for sharing MATLAB programs with end user.

3.2 MATLAB PROGRAMMING

➢ Basic setup :

Setting up MATLAB environment is a matter of few clicks. MathWorks provides the licensed product, a trial version and a student version as well. You need to log into the site and wait a little for their approval. After downloading the installer the software can be installed through few clicks.

After logging into your account, you can enter MATLAB by double-clicking on the MATLAB shortcut icon on your Windows desktop. When you start MATLAB, a special window called the MATLAB desktop appears. The desktop is a window that contains other windows.

The major tools within or accessible from the desktop are:

- The Command Window
- The Command History
- The Workspace
- The Current Directory
- The Help Browser
- The Start button

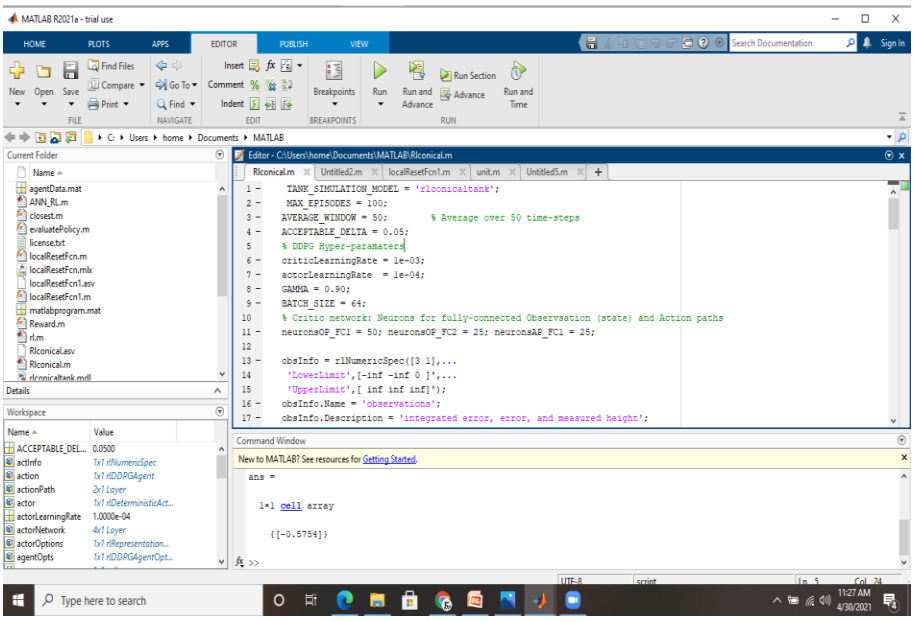


Fig 3.1 MATLAB IDE

- **Managing the workspace** :

The contents of the workspace persist between the executions of separate commands. Therefore, it is possible for the results of one problem to have an effect on the next one. To avoid this possibility, it is a good idea to issue a clear command at the start of each new independent calculation.

- **Syntax**

  >> clear

The command clear or clear all removes all variables from the workspace. This frees up system memory. In order to display a list of the variables currently in the memory, type

- **Syntax**

  >> who

while, whos will give more details which include size, space allocation, and class of the variables.

- **Keeping track of your work session** :

It is possible to keep track of everything done during a MATLAB session with the diary command.

- **Syntax:**

  >> diary

  or give a name to a created file,

- **Syntax**

  >> diary FileName

. where FileName could be any arbitrary name you choose. The function diary is useful if you want to save a complete MATLAB session. They save all input and output as they appear in the MATLAB window. When you want to stop the recording, enter diary off. If you want to start recording again, enter diary on. The file that is created is a simple text file.. You can use the function type to view the diary file or you can edit in a text editor or print. This command is useful, for example in the process of preparing a homework or lab submission

➢ Getting help :

To view the online documentation, select MATLAB Help from Help menu or MATLAB Help directly in the Command Window. The preferred method is to use the Help Browser. The Help Browser can be started by selecting the ? icon from the desktop toolbar. On the other hand, information about any command is available by typing

- **Syntax**

  >> help Command

Another way to get help is to use the lookfor command. The lookfor command differs from the help command. The help command searches for an exact function name match, while the lookfor command searches the quick summary information in each function for a match. For example, suppose that we were looking for a function to take the inverse of a matrix. Since MATLAB does not have a function named inverse, the command help inverse will produce nothing. On the other hand, the command lookfor inverse will produce detailed information, which includes the function of interest, inv.

- **Syntax** :

  >> lookfor inverse

➢ Basic plotting:

MATLAB has an excellent set of graphic tools. Plotting a given data set or the results of computation is possible with very few commands. You are highly encouraged to plot mathematical functions and results of analysis as often as possible. Trying to understand mathematical equations with graphics is an enjoyable and very efficient way of learning mathematics. Being able to plot mathematical functions and data freely is the most important step, and this section is written to assist you to do just that.

- **Syntax**

  >> plot(x,y)

➢ Matrix Generation:

Matrices are the basic elements of the MATLAB environment. A matrix is a two-dimensional array consisting of m rows and n columns. Special cases are column vectors (n = 1) and row vectors (m = 1). In this section we will illustrate how to apply different operations on matrices. The following topics are discussed: vectors and matrices in MATLAB, the inverse of a matrix, determinants, and matrix manipulation. MATLAB supports two types of operations, known as matrix operations and array operations. Matrix operations will be discussed first.

A vector is a special case of a matrix. The purpose of this section is to show how to create vectors and matrices in MATLAB. As discussed earlier, an array of dimension 1 ×n is called a row vector, whereas an array of dimension m × 1 is called a column vector. The elements of vectors in MATLAB are enclosed by square brackets and are separated by spaces or by commas. For example, to enter a row vector, v, type

- **Syntax**

  >> v = [1 4 7 10 13]

➢ Entering a matrix:

A matrix is an array of numbers. To type a matrix into MATLAB you must

- begin with a square bracket, [
- separate elements in a row with spaces or commas (,)
- use a semicolon (;) to separate rows
- end the matrix with another square bracket,].

➢ Scrolling

Suppose you want to repeat or edit an earlier command. If you dislike typing it all back in, then there is good news for you: MATLAB lets you search through your previous commands using the up-arrow and down-arrow keys. This is a very convenient facility, which can save a considerable amount of time.

➢ Keeping in touch with your variables

By now you might have lost track of what variables you created. Yet they are still there as we

mentioned before: Unless you explicitly delete them, change them or quit MATLAB, they do not go away, but are kept in MATLABs workspace. You can see a list of all your variables in the Workspace window. Alternatively you can use commands in the Command window to do the same:

1) Type **who** to see all your matrices.

2) Type **whos** to see a more complete description, including sizes, of all your stored data. From this description check that v is a row vector and c is a column vector.

➢ Generating variables made easy

Ready-made matrices and vectors

MATLAB provides functions to create several basic matrices automatically without having to type or read in each of the elements. The most important functions are

- **zeros** **zeros(m,n)** creates an mxn matrix whose elements are equal to zero
- **ones** **ones(m,n)** creates an mxn matrix whose elements are equal to one.
- **eye** **eye(m,n)** creates an mxn identity matrix
- **rand** **rand(m,n)** creates an mxn matrix whose elements are all random number between 0 and 1

➢ Changing values of individual elements

Individual elements in a variable can be identified using index numbers. For example the element 3 in the matrix A above can be referred to as A(2,1) because it is the first element of the second row of A. Note that the row and column indices are separated by a comma.

1) Change this element to have the value of 1 by typing **A(2,1)=1**

Note how MATLAB prints the whole matrix (unless you include a ; at the end), and that A(2,1) has changed. Similarly sections of matrices (sub-matrices) can be identified using the colon operator.

➢ Saving and retrieving your work

All variables that you created in this MATLAB session are stored in MATLAB's workspace. Upon exiting MATLAB this workspace will be destroyed. So you must save your workspace if you want to use it in later MATLAB sessions.

1) Select *Save Workspace As* under *File* on the menu. A save window will come up. Check that the correct folder is given. Type in the name you wish to use for the workspace file, for example 'lab6' and save the file. MATLAB automatically selects the '.mat' extension for the workspace files.

- **VERY IMPORTANT:**

These .mat files are **BINARY** files which means that you CAN NOT EDIT OR READ THEM like you can with ascii files. SO DO NOT ATTEMPT TO DO THAT. Reading the saved information MUST be done using the MATLAB *Import Data* option.

2) Type **clear** to delete your workspace. Type **who** to check that your workspace is empty (or check your Workspace window). Retrieve the saved workspace using *Import Data* under *File*. Type **who** again to check that all your variables are back in the workspace (or check the Workspace window). So, use the *Import Data* option to load mat files back into memory.This method of saving saves <u>all</u> the variables in your workspace. You can select any number of variables if you save from within the Command Window instead.

3) Create two random variables by typing **x=rand;** and **y=rand;** To save these two variables in a .mat file called 'savexy.mat' type **save savexy x y** .The contents of your current directory is shown in the Current Directory window. But you can also find out its contents using a command:

4) The command **dir** gives you the list of all files that are in your current directory. The command **what** gives you the files in the current folder relevant to MATLAB only.

➢ Script files

In this section we show you how to store commands in a file. Such MATLAB command files are

known as *script files*. They are stored with the .m extension. In MATLAB files with the .m extension are also called M-files. All script files are M-files, but not all M-files are script files as you will see later (for example, MATLAB functions are also stored in .m files)

➢ Running a program step-by-step, one command at a time

MATLAB runs the addmat program without breaks or pauses. Often it is convenient to run a program step by step to carefully check that all the lines of code that you wrote are doing what you expect them to do. This is a good way to find errors in your code, or in other words to debug your code.

➢ Logical expressions

• Relational operators

MATLAB has 6 relational operators to make comparisons between variables. These are

<	is less than
< =	is less than or equal to
>	is greater than
> =	is greater than or equal to
= =	is equal to
~ =	is not equal to

Note that the *is equal to* operator consists of two equal signs and not a single = sign as you might expect. We'll play around with these relational operators for a little while.

1) Go back to the Command Window. Set up the scalars q=10, w=10, and e=20.

In MATLAB a logical expression has two possible values that are not 'true' or 'false' but *numeric*, i.e. 1 if the expression is true and 0 if it is false.

2) Type **w< e** to see that the result is given the value of 1 because w is indeed less than e. Now type

q==e. The == operator checks if two variables have the same value. Therefore the answer is 0 in this case.

Relational operations are performed *after* arithmetic operations. For example **q == w-e** results in 0. Parentheses can be used to override the natural order of precedence. So **(q==w) - e** results in -19. The relational operators can be used for all elements of vectors simultaneously.

- **Logical Operators**

For **&** (and) to give a true result both expressions either side of the **&** must be true. The logical expression , **e> 0** is true and the logical expression **q<0** is false. So the logical expression (**e>0) & (q<0**) is false. For **|** (or) to give a true result only one of the expressions either side of the **|** needs to be true. Type (**e>0) | (q<0)** .

The **~** (not) operator changes a logical expression from 0 to 1 and vice versa. So **result = ~(q<0)** would be 1. The operator ~ has a high priority. So, to avoid mistakes, put whatever expression you want to negate in parentheses, as we did here. The operations **&** and **|** are performed after the relational operations,< > = = etc. Just as with relational operators we can use logical operators for vectors or matrices.

➢ The 'if' statements

The if-statement starts with **if** followed by a condition, usually a relational operation, and is finished with an
end; command. For example (taken from a soccer game simulator)

```
if score ~= 0

disp('GOAL – the score is ');

disp(score);

end;
```

The body of this if-statement consists of two lines and both will be carried out if the score is not equal to zero. We can include as many lines in a body as we want.

3.3 MATLAB SIMULINK :

Simulink is a graphical extension to MATLAB for modeling and simulation of systems. In Simulink, systems are drawn on screen as block diagrams. Many elements of block diagrams are available, such as transfer functions, summing junctions, etc., as well as virtual input and output devices such as function generators and oscilloscopes. These virtual devices will allow you to perform simulations of the models you will build.

Simulink is integrated with MATLAB and data can be easily transferred between the programs. In this tutorial, we will apply Simulink to the examples of modeled systems, then build controllers, and simulate the systems

➢ Starting Simulink

Simulink is started from the MATLAB command prompt by entering the following command:

```
Simulink
```

When it starts, Simulink brings up two windows. The first is the main Simulink window, which appears as shown or similar to this as different versions of the software are found:

➢ Model Files:

In Simulink, a model is a collection of blocks which, in general, represents a system. In addition, to drawing a model into a blank model window, previously saved model files can be loaded either from the **File** menu or from the MATLAB command prompt. As a first exercise we will be building a model.

You can open a file in Simulink by entering the following command in the MATLAB command window. (Alternatively, you can load this file using the **Open** option in the **File** menu in Simulink, or by hitting

Ctrl+O in Simulink.)

```
File name.mdl
```

For our purposes we will create the following model in simulink. A new model can be created by selecting **New** from the **File** menu in any Simulink window (or by hitting Ctrl+N). Using the information below, create the following model.

➢ Basic Elements

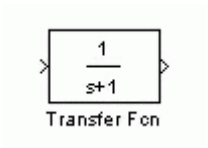
There are two major classes of items in Simulink: **blocks** and **lines**. Blocks are used to generate, modify, combine, output, and display signals. Lines are used to transfer signals from one block to another.

• Blocks

There are several general classes of blocks:

- Sources: Used to generate various signals
- Sinks: Used to output or display signals
- Discrete: Linear, discrete-time system elements (transfer functions, state-space models, etc.)
- Linear: Linear, continuous-time system elements and connections (summing junctions, gains, etc.)
- Nonlinear: Nonlinear operators (arbitrary functions, saturation, delay, etc.)
- Connections: Multiplex, Demultiplex, System Macros, etc.

Blocks have zero to several input terminals and zero to several output terminals. Unused input terminals are indicated by a small open triangle. Unused output terminals are indicated by a small triangular point. The block shown below has an unused input terminal on the left and an unused output terminal on the right.



➢ Introduction to MATLAB graphics

MATLAB has a large number of functions associated with graphical output. If you'd like to explore the possibilities use **help plot** or **help plot3** for 3-dimensional plots, or run the MATLAB demo (by typing **demo**) and look at the information on visualization and graphics.

We start with basic plotting routines and look at some fancy graphics to get a taste of MATLAB's abilities.

- **Lines**

  Lines transmit signals in the direction indicated by the arrow. Lines must always transmit signals from the output terminal of one block to the input terminal of another block. On exception to this is a line can tap off of another line, splitting the signal to each of two destination blocks, as shown. Lines can never inject a signal *into* another line; lines must be combined through the use of a block such as a summing junction.

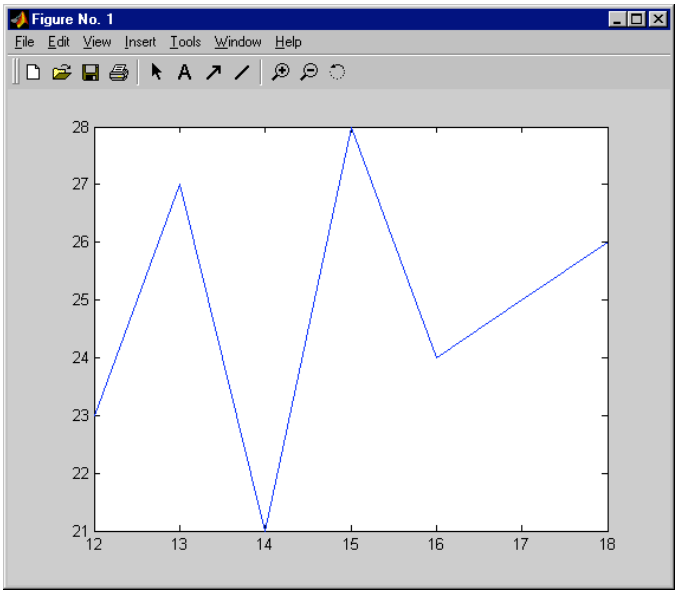


Fig 3.2 Line Plot

A signal can be either a scalar signal or a vector signal. For Single-Input, Single-Output systems, scalar signals are generally used. For Multi-Input, Multi-Output systems, vector signals are often used, consisting of two or more scalar signals. The lines used to transmit scalar and vector signals are identical. The blocks on either end of the line determine the type of signal carried by the line.

- **Modify Blocks**

Follow these steps to properly modify the blocks in your model.

- Double-click your Sum block. Since you will want the second input to be subtracted, enter +- into the list of signs field. Close the dialog box.
- Double-click your Gain block. Change the gain to 2.5 and close the dialog box.
- Double-click the leftmost Transfer Function block. Change the numerator to [1 2] and the denominator to [1 0]. Close the dialog box.
- Double-click the rightmost Transfer Function block. Leave the numerator [1], but change the denominator to [1 2 4]. Close the dialog box.

- **Taking Variables from MATLAB**

In some cases, parameters, such as gain, may be calculated in MATLAB to be used in a Simulink model. If this is the case, it is not necessary to enter the result of the MATLAB calculation directly into Simulink. For example, suppose we calculated the gain in MATLAB in the variable K. Emulate this by entering the following command at the MATLAB command prompt.

```
K=2.5
```

This variable can now be used in the Simulink Gain block. In your simulink model, double-click on the Gain block and enter the following in the Gain field.

```
K
```

➢ Exchanging Signals with MATLAB

Sometimes, we would like to use the results of a Simulink simulation in the MATLAB command window for further calculations and plotting. Less often, we would like to generate signals in MATLAB which we then use as inputs in a Simulink model. Workspace Source Block. We will only transfer signals from Simulink to MATLAB. Doing the reverse is a very similar process.

➢ Extracting Models From Simulink into MATLAB

Sometimes, we may build a complicated model in simulink and would like to derive either a transfer function or a state space model of the entire system. In order to do this, you first need to define the input and output signals of the model to be extracted. These virtual signals can be any signal in a model, for example, if we can generate an input-to- output transfer function or a disturbance-to-error transfer function. These signals are defined using the In and Out Connection Blocks.

Once the input/output model is defined, the Simulink model must be saved to a .mdl file. This file is then referenced in the MATLAB command window by the *linmod* command.

➢ Reinforcement Learning Toolkit:

Reinforcement Learning Toolbox™ provides functions and blocks for training policies using reinforcement learning algorithms including DQN, A2C, and DDPG. You can use these policies to implement controllers and decision-making algorithms for complex systems such as robots and autonomous systems. The toolbox lets you train policies by enabling them to interact with environments represented by MATLAB® or Simulink® models. You can evaluate algorithms, experiment with hyperparameter settings, and monitor training progress. Through the ONNX™ model format, existing policies can be imported from deep learning frameworks such as TensorFlow™ Keras and PyTorch (with Deep Learning Toolbox™). You can generate optimized C, C++, and CUDA code to deploy trained policies on microcontrollers and GPUs. The toolbox includes reference examples for using reinforcement learning to design controllers for robotics and automated driving applications.

3.4 Reinforcement Learning in MATLAB:

The deep deterministic policy gradient (DDPG) algorithm is a model-free, online, off-policy reinforcement learning method. A DDPG agent is an actor-critic reinforcement learning agent that searches for an optimal policy that maximizes the expected cumulative long-term reward. DDPG agents can be trained in environments with the following observation and action spaces During training, a DDPG agent:

- Updates the actor and critic properties at each time step during learning.

- Stores past experiences using a circular experience buffer. The agent updates the actor and critic using a mini-batch of experiences randomly sampled from the buffer.
- Perturbs the action chosen by the policy using a stochastic noise model at each training step.

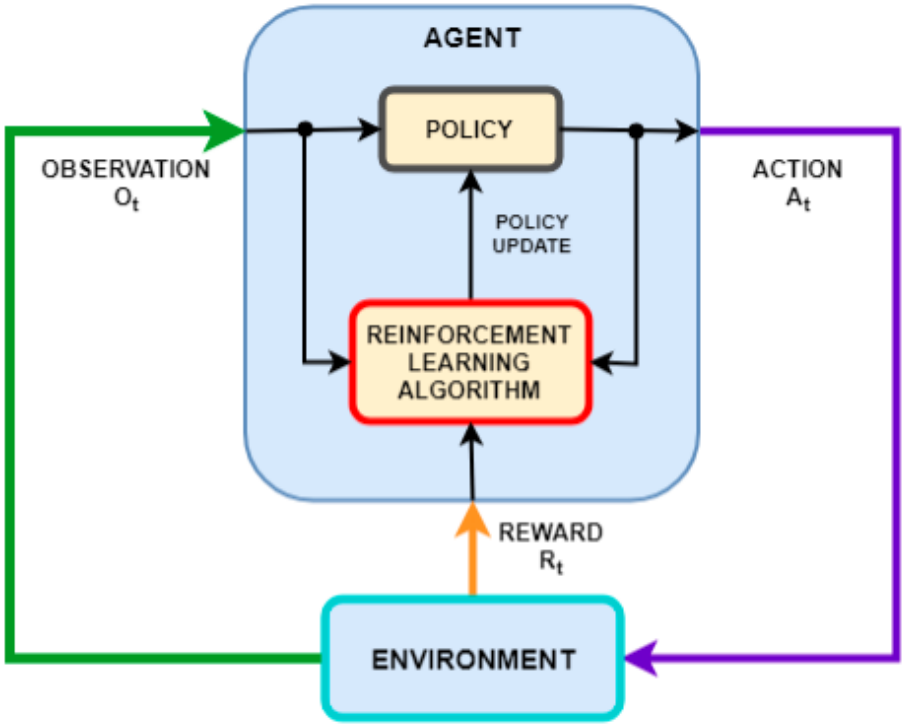


Fig 3.3 RL Block Diagram

➢ Actor and Critic Functions

To estimate the policy and value function, a DDPG agent maintains four function approximators:

- Actor $\mu(S)$ — The actor takes observation S and returns the corresponding action that maximizes the long-term reward.
- Target actor $\mu'(S)$ — To improve the stability of the optimization, the agent periodically updates the target actor based on the latest actor parameter values.
- Critic $Q(S,A)$ — The critic takes observation S and action A as inputs and returns the corresponding expectation of the long-term reward.
- Target critic $Q'(S,A)$ — To improve the stability of the optimization, the agent periodically updates the target critic based on the latest critic parameter values.

Both $Q(S,A)$ and $Q'(S,A)$ have the same structure and parameterization, and both $\mu(S)$ and $\mu'(S)$ have the same structure and parameterization.

When training is complete, the trained optimal policy is stored in actor $\mu(S)$.

3.5 CONCLUSION:

To conclude, MATLAB comes complete with a huge library of predefined functions that provides tested and pre-packaged solutions to many primary technical tasks. In addition to the vast libraries of services built into the basic MATLAB language, MATLAB R2021a version is chosen as there are many special-purpose toolboxes applicable to help solve complex problems in particular areas. For example, a user can buy standard toolkits to solve problems in signal processing, control systems, communications, image processing, and neural networks, etc. Also, Simulink updates enable users to import C code as reusable Simulink libraries and to speed up simulations. One such toolbox is imported into our model which is Reinforcement learning toolbox. This Can be applied in Optimizing many non linear systems.

CHAPTER 4

HARDWARE DESCRIPTION

This chapter explains about major hardware used in Conical tank system and. It gives detailed description of each of the above, working, specifications, and advantages.

4.1 PUMP

A pump is a device that moves fluids (liquids or gases), or sometimes slurries, by mechanical action, typically converted from electrical energy into hydraulic energy. Pumps can be classified into three major groups according to the method they use to move the fluid: direct lift, displacement, and gravity pumps.

Pumps are commonly rated by horsepower, volumetric flow rate, outlet pressure in meters (or feet) of head, inlet suction in suction feet (or meters) of head. The head can be simplified as the number of feet or meters the pump can raise or lower a column of water at atmospheric pressure.

Efficiency: Pump efficiency is defined as the ratio of the power imparted on the fluid by the pump in relation to the power supplied to drive the pump. Its value is not fixed for a given pump, efficiency is a function of the discharge and therefore also operating head. For centrifugal pumps, the efficiency tends to increase with flow rate up to a point midway through the operating range (peak efficiency or Best Efficiency Point (BEP)) and then declines as flow rates rise further. Pump performance data such as this is usually supplied by the manufacturer before pump selection.

Fig.4.1 Pump

➢ Specifications of Pump :

Company	Kirloskar/ Equivalent
Head	3 to 5 meters
Discharge	up to 8 meters
Supply Voltage	230VAC/50 HZ
Power	0.5 HP
Speed	1450 RPM
Flow Rate	1050 LPH
End Connection	1” screwed

Table 4.1 : Specifications of Motor pump.

➢ Types of Pumps:

1. **Positive-displacement pumps** :A positive-displacement pump makes a fluid move by trapping a fixed amount and forcing (displacing) that trapped volume into the discharge pipe.Some positive-displacement pumps use an expanding cavity on the suction side and a decreasing cavity on the discharge side. Liquid flows into the pump as the cavity on the suction side expands and the liquid flows out of the discharge as the cavity collapses. The volume is constant through each cycle of operation.
2. **Centrifugal pumps :**Centrifugal pumps are used to transport fluids by the conversion of rotational kinetic energy to the hydrodynamic energy of the fluid flow. The rotational energy typically comes from an engine or electric motor. They are a sub-class of dynamic axisymmetric work-absorbing turbomachinery.The fluid enters the pump impeller along or near to the rotating axis and is accelerated by the impeller, flowing radially outward into a diffuser or volute chamber (casing), from which it exits.Common uses include water, sewage, agriculture, petroleum and petrochemical pumping. Centrifugal pumps are often chosen for their high flow rate capabilities, abrasive solution compatibility, mixing potential, as well as their relatively simple engineering
3. **Axial Flow pumps :**An axial-flow pump, is a common type of pump that essentially consists of a propeller (an axial impeller) in a pipe. The propeller can be driven directly by a sealed motor in the pipe or by electric motor or petrol/diesel engines mounted to the pipe from the outside or by a right-angle drive shaft that pierces the pipe.An axial flow pump has a propeller-type of impeller running in a

casing. The pressure in an axial flow pump is developed by the flow of liquid over the blades of impeller.

4.2 CONTROL VALVE:

A control valve is a valve used to control fluid flow by varying the size of the flow passage as directed by a signal from a controller. This enables the direct control of flow rate and the consequential control of process quantities such as pressure, temperature, and liquid level.

A control valve is a power-operated device used to regulate or manipulate the flow of fluids, such as gas, oil, water, and steam.It is a critical part of a control loop and is an example of a final control element. The Control Valve is by far the most common final control element used in industry today.

➢ Introduction to Pneumatic Control Valve:

A valve in which the force of compressed air against a diaphragmis opposed by the force of a spring to control the area of opening for a fluid steam. It consists of an Actuator and a valve. The actuator moves the valves stem as the pressure on a spring-loaded diaphragm changes.Actuatorconvertscontroller output signal (4-20 mA or 3-15psgi) to physical adjustment in the process input variables.

Fig.4.2.1 Control Valve

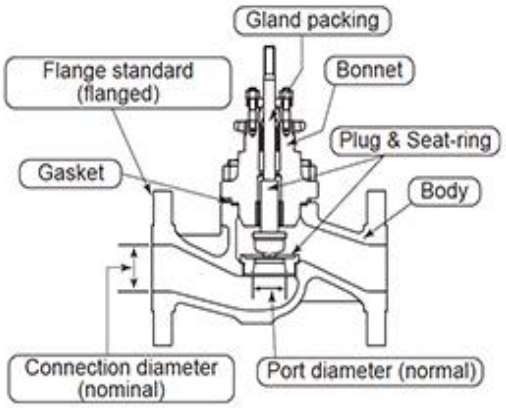


Fig.4.2.2 Internal structure

➢ Specifications :

Company	RK Control Valves
Type	Single Seated Globe Control Valve
Characteristics	Equal %
Action	Air failure to close
Body Material	Carbon Steel
Trim Material	Stainless Steel
Valve Co-efficient	5
Stem Travel	(0 – 12)mm
Control Signal	(3 – 15) PSI
End Connection	3/4" flanged

Table 4.2: Specifications of Control Valve

4.3 ROTAMETER

A rotameter is a device that measures the volumetric flow rate of fluid in a closed tube. It belongs to a class of meters called variable-area flowmeters, which measure flow rate by allowing the cross-sectional area the fluid travels through to vary, causing a measurable effect. A rotameter (variable area meter) is a flow meter that measures volumetric flow of liquids and gases. There is no difference between a rotameter and flow meter, and these terms are used interchangeably. The technique for measuring flow is accomplished by a freely moving float finding equilibrium in a tapered tube.

➢ Working principle:

Rotameter works on the principle of upthrust force exerted by fluid and force of gravity. The buoyant force exerted on an immersed object is equal to the weight of liquid displaced by the object. Under this principle, the rotameter works with float-tapered tube system.

➢ Construction:

A rotameter is made up of a tapered tube and a float inside it. The glass tapered tube has a scale on the surface or a scale is placed adjacent to it, according to purpose.

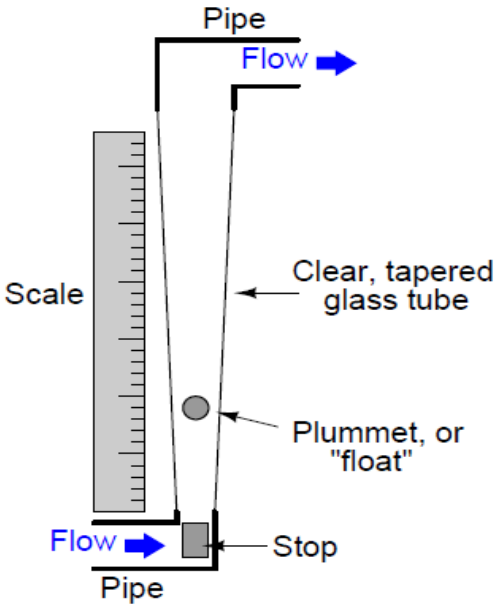


Fig.4.3.1 Rotameter

- **Tapered tube:**

The tapered tube is placed vertically in the flow channel with a conical shape inside. The quantity measured is defined by the height of float going up. Glass tubes are used for both liquid and gas measurement. Metallic tubes are used where the process fluid with high temperature and pressure.

- **Float:**

Stainless steel floats are commonly used, there are different types of metals from lead to aluminium used as floats. A float material, shapes are also varied according to applications considering density.

➢ Working:

Fluid enters from the bottom of the tapered tube, then some of the fluid strikes directly into the float bottom and others pass aside the float. Now the float experience two forces in opposite direction, darg force upward and gravitational force downward.

Fluid flow moves the float upward against gravity. At some point, the flowing area reaches a point where the pressure-induced force on the floating body exactly matches the weight of the float. The float will find equilibrium when the area around float generates enough drag equal to weight - buoyancy.

As the float weight and gravity are constant, the distance float displaced upward is proportional to the flow velocity of the fluid passing through the tapered tube.

➢ Measuring Principles of Variable Area Flowmeters:

$$Q = k\sqrt{\frac{P_1 - P_2}{\rho}}$$

However, the difference in this application is that the value inside the radicand is constant since the pressure difference will remain constant and the fluid density will likely remain constant as well. Thus, k will change in proportion to Q. The only variable within k relevant to float position is the flowing area between the float and the tube walls.

➢ Specifications :

Company	Sun Flow/ Equivalent
Type	Variable Area
Range	upto 1000 LPH
Float material	Stainless Steel
Tube material	Glass
Body material	Mild steel with Powder Coating
End connection	¾” SS Flanged

Table 4.3 Specifications of Rotameter

➢ Applications :

- The rotameter is used in process industries.
- It is used for monitoring gas and water flow in plants or labs.
- It is used for monitoring filtration loading.

Fig. 4.3.2 Rotameter Device

➢ Advantages of rotameter:

- The cost of rotameter is low.
- It provides linear scale.
- It has good accuracy for low and medium flow rates.
- The pressure loss is nearly constant and small.
- Usability for corrosive fluid.

➢ Disadvantages of rotameter:

- When opaque fluid is used, float may not be visible.
- It has not well in pulsating services.
- Glass tube types subjected to breakage.
- It must be installed in vertical position only.

4.4 I/P CONVERTER:

An I/P converter is a current-to-pressure transducer utilized in industrial control systems. It is a small module device used in applications to translate a current analog signal (I) into a pneumatic output (P). In other words, it takes electrical signals from a DC (direct current) signal and converts those signals into proportional pneumatic pressure. I/P transducers convert electrical signals from a controller to apply pneumatic pressure to valves, dampeners, actuators, or brakes and clutches within an industrial mechanism.

Before we go into the details of an IP converter it is important to note that typically there are two types of

these converters available, an IP converter and an EP converter. The IP converter most commonly receives a 4-20mA signal and converts it to a pneumatic output. The difference with an EP converter is that it receives a 0-10v signal and coverts it to a pneumatic output. The IP converter works with current and the EP converter works with voltage.

➢ IP Converter Working Principle:

I/P Converters operate on a similar principle as proportional valves in that they provide adjustable flow volumes and control functions. In converting electric current into pneumatic output, I/P converters also eliminate a need for an external power supply to control the pneumatic operated components. In doing so, I/P converters increase machine efficiency and control.

The function of I/P converters relates to the concept of an electromagnetic force balance principle, i.e., the process of converting electric current into force and then pressure. Simply, for I/P converters to work, electromagnetic effects are exploited to build current which force transducers for mechanical actuation. This action converts electrical current (typically 4 to 20 mA) into pneumatic pressure output (3 to 15 psi) that, in turn, proportionally controls the opening or closing of a valve according to the force balance principle.

As I/P converters operate, they deliver proportional downstream flow of pressure related to the rate of electrical current that is received into the transducer's electrical circuit. In web tension control systems, I/P converters serve as an important connecting interface between the controller and pneumatic brakes and clutches.

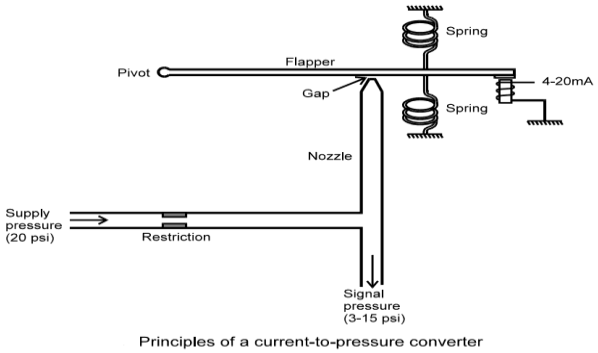


Fig.4.4 I/P Internal Structure

An I/P converter operates on the electromagnetic force balance principle. They are an important component in maintaining consistent web tension control. I/P converters must not only regulate the electrical current signal from the controller and convert it into a regulated pneumatic control signal, i.e., into a proportional pneumatic

output, but must also be able to do so accurately with dependability when converting signals. For web tension control, I/P converters offer the necessary high flow rate to provide the rapid response required to accurately control either the dancer or load cell tension control system.

- **Specifications**:

Company	Fairchild/ Equivalent
Input Signal	(4 – 20)Ma @ 20 PSI Air
Output	(3 – 15) PSI
End Connection	¼" BSP
Accuracy	0.5% FS
OperatingTemperature	-20 to 70°C (-5 to 160°F)
InputResistance	<300 Ω
FlowRate	10 scfm ≤60 psi; 0.06 <60 psi
FailureMode	Upon electrical failure, the signal pressure falls to bleed pressure
PressurePort	¼ FNPT
ElectricalConnection	DIN 43650 with screw terminals included
Housing	IP65 rated, epoxy-painted zinc die castings

Table 4.4 Specifications of I/P Converter

- **I/P Converter and its types:**

Proportional valves from Norgren and I/P Converters from Watson Smith all possess different features, making them suitable for individual applications. Below, we look at the different features of current to pressure converters and the potential workings of an I/P converter.

- Intrinsically safe I/P converter

  An intrinsically safe I/P converter is built to work effectively in industries where flammable gases or liquids may be applied. Intrinsically safe I/P converter uses include natural gas and ATEX environments.

- ATEX Proportional valve

  If a component is described as being ATEX certified, it can be used in any work environment that may be at risk of explosion. Therefore, ATEX I/P Converters applications mostly include dangerous environments, where hazardous or reactive chemicals are used.

- Fail freeze I/P Converter

  If an I/P converter features fail freeze, it means the component will fail in its latest position in cases of loss of electronic signal. Therefore, typical I/P converter uses include damper applications, such as boilers and fans.

➢ APPLICATIONS:

The most common application of an I/Ptransducer is to receive an electrical signal from a controller and produce a proportional pneumatic output for operating a control valve or positioner. The device can be mounted on the wall or a pipe stand or directly on the valve actuator.

Current to pressure converters are suitable for a wide range of applications, as they boast high stability, low maintenance and a wide range of air supply options. Their array of features means an I/P Converter can be applied in both demanding, natural gas applications and more general industry use. I/P converter applications largely depend on the features of the proportional valve. For example, some proportional valves may be suited to general purpose applications, while others can be applied in hazardous environments.

➢ I/P converter calibration:

Generally, I/P converter is calibrated for standard industrial signals as 4-20 mA input and 3-15 psi output.

The following steps to calibrate I/P converter:

1. Remove the I/P from the process.
2. If there is any top cover in I/P remove it.
3. Connect 20-psi supply pressure and connect a milliamp source to the I/P .
4. Set the input signal to 4 mA and check the output pressure on gauge as 3 psi.
5. If the pressure is showing more or less than 3 psi then adjust zero.

6. Turn zero adjustment screw slowly by very small turn to obtain 3-psi pressure. More turning of zero adjustment may damage the I/P converter. Counterclockwise rotation increases the pressure, and clockwise rotation decreases the pressure.
7. Set the input current signal to 20 mA and check the output pressure on gauge as 15 psi.
8. Turn the span adjustment potentiometer very slowly by small turn to obtain 15-psi pressure. More turning of span adjustment may damage the I/P converter.

4.5 LEVEL TRANSMITTER

A Level Transmitter is simply an instrument that provides continuous level measurement. Level transmitters can be used to determine the level of a given liquid or bulk-solid at any given time. This is different to a level switch which only alarms when the level of material (liquid or bulk-solid) reaches a predetermined level.Level sensors detect the level of liquids and other fluids and fluidized solids, including slurries, granular materials, and powders that exhibit an upper free surface.Level transmitters provide continuous level measurement over the range of the system rather than at a single point and produce an output signal that directly correlates to the level within a vessel. From depth and pressure, to float, radar, ultrasonic, and capacitive, these transmitters employ different technologies designed individually for specific level applications.

Fig.4.5.1 Level Transmitter

➢ Working Principle of Level Transmitters:

The working principle of level transmitters mentioned above varies according to their underlying principle. For instance, capacitance level transmitters operate through a capacitor, hydrostatic level transmitters depend on the pressure of a fluid in a storage container for level measurement, while ultrasonic level transmitters convert the distance travelled by an ultrasonic wave to determine the level, and so on. However, all these level

transmitters measure the level in either of the three ways:

- The weight of the fluid
- The pressure head of the fluid
- The position of the fluid in a container

If you look closely, all the pressure transmitters covered in this post take into account either of the three factors to give an appropriate measurement. Level measurements are classified into two types– direct and indirect level measurements or performed by contact or non-contact transmitters. Direct level measurements are considered ideal for small level changes, which are observed in various industrial tanks. However, most level transmitters are designed for indirect level measurements as they are sensitive and designed for too high or too low level measurements, where direct measurement becomes difficult. Ultrasonic level transmitters can be availed in contact- or non-contact configurations.

These transmitterswork on the principle of a radar by using radio wave emissions. The transmitter sends a radar signal into the liquid and receives a reflection of the signal. The transmitters then analyse the current fill level of the tank based on the time taken by the transmitted signal to return

➢ Types of Level Transmitters:

Level measurement transmitters are of seven types. Each type of transmitter works in a different way and makes it useful for different types of processes.

- ### Capacitance Level Transmitters

  These transmitters use liquid stored in a tank or container as a dielectric medium between two or more electrodes. The energy capacity of the capacitor circuit increases when there is more liquid and decreases if there is less liquid. By measuring the variations in the capacitance value, capacitance level transmitters can calculate the current fill level of the tank or container.

- ### Hydrostatic Level Transmitters

  Also known as pressure level transmitters, these transmitters help in determining fluid contents of a container by measuring the pressure of resting body of the fluid within it. The greater the force of liquid, the greater the volume of fluid.

- **Magnetic Level Transmitters**

  These transmitters use a magnetic object, which is suspended in a buoyant float. This is usually in a narrow auxiliary column, to restrict lateral movements of the float. While the float is on top of the liquid, the movement of the float is measured by a different magnetic device. This allows a precise and stable fill level to be transmitted. This method is suitable for continuous measurement owing to the tendency of the float to rise or sink based on the liquid level.

- **Radar Fill Level Transmitters**

  These transmitters work on the principle of a radar by using radio wave emissions. These transmitters are normally mounted at the top of a tank filled with a liquid. The transmitter sends a radar signal into the liquid, and receives a reflection of the signal. The transmitters then analyze the current fill level of the tank based on the time taken by the transmitted signal to return.

- **Ultrasonic Level Transmitter**

  In this type of transmitter, an ultrasonic transducer is mounted at, or near the top of a container containing liquid. The transducer sends out an ultrasonic pulse. The pulse hits the surface of the liquid, and gets reflected. The sensor then calculates the fill level based on the time between the transmitted and received signal.

- **Guided Microwave Level Transmitters**

  These transmitters work by sending a microwave pulse through a sensor cable or rod. The signal hits the surface of the liquid, and travels back to the sensor, and then to the transmitter housing. The electronics integrated in the transmitter housing determine the filling level based on the time taken by the signal to travel down the sensor and back up again. These types of level transmitters are used in industrial applications in all areas of process technology.

- **Liquid Level Transmitters**

  These transmitters are engineered for detecting levels of liquids. Liquid level transmitters are also used for detecting interfaces between two different liquids such as oil and water. Liquid level transmitters are mainly used for liquid-level sensing in storage tanks, transport tanks, as well as water storage tanks. These pressure transmitters measure the level by measuring the head pressure of liquid.

➢ Specificities of Level measurement with differential pressure transmitter:

Measurement of level in industrial application by differential pressure transmitter is frequent. The DP Transmitter device is used to measure level as an inferential measurement. In a DP Transmitter, the diaphragm senses the head pressure developed by the height of the material in the vessel.This quantity is multiplied by a density variable to get the correct level measurement. We have listed below some unique specialty of DP Transmitter for level measurement.

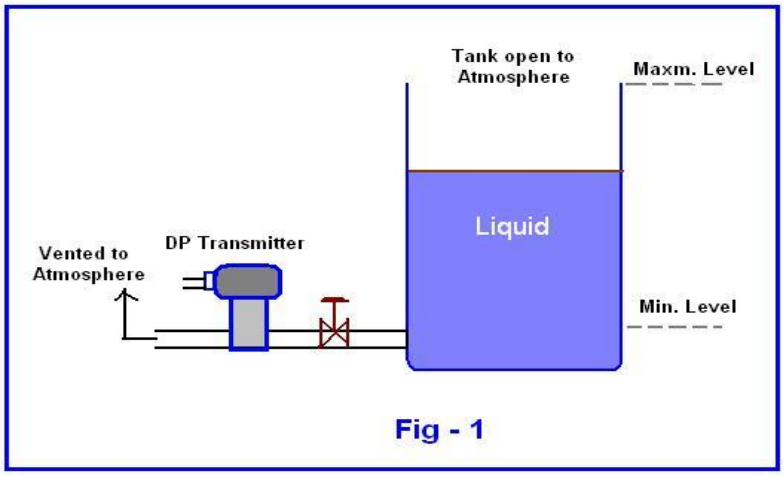


Fig.4.5.2 Internal Structure

- A great specificity of Differential Pressure Transmitters is that it is effortlessly fitted to an existing vessel. It can also be retrofitted to a working Tank or Vessel. Maintenance of DP Transmitter is easy as it can be isolated from the process by isolation valve. It is ideal and may be the only alternative for total level measurement in separator vessels, because separator vessel undergoes a wide variation of composition in process materials.

- DP transmitters are ideal for level measurements of liquids. It can also be used for light slurries with extended diaphragm, which fits flush, to one side of the vessel.
- Level measurement by DP Transmitters is more cost effective than other available sensors.

- A broad range of accurate level can be measured by differential pressure transmitters subject to the stability of the fluid density. In case of unstable process fluid density, an extra DP Transmitter is used to estimate the density.

- Since DP Transmitters are located away from the tank for measurement of level, cleaning and maintenance of Tanks are easy.

- Accumulation of water at the bottom of a tank can be ascertained by Differential Pressure Transmitters.

- When fluid density changes, the multivariable transmitter, can measure tank level accurately. It will deliver the same accuracy in both open or pressurized tanks and vessels. The technology responsible for this improved density compensated level transmitter is due to hydrostatic measurement and sophisticated multivariable sensing technology. The transmitter uses onboard, uninterrupted differential pressure, and temperature measurements to offset for liquid and vapor density changes. The final data are transmitted in 4 to 20mA protocol. Digital output signals are accurately proportional to tank level.

Company	ABB/Equivalent
Type	DPT with SMART and HART
SensorType	Diaphragm(capacitive)
Range	Up to 4000mmwc
SupplyVoltage	24v DC@ 150MA
Output	(4 – 20) MA

Table 4.5 Specifications of Level Transmitter

4.6 CONICAL TANK

Conicaltank storage containers are cone-shaped vessels used for storage of water, chemicals, fertilizers, etc. They have a wide variety of applications and are upside down to facilitate easy outflow of liquids and forbid contaminant accumulation.

Features of Conical Tanks:

- FDA Compliant Material
- Resistance to chemicals and corrosive acids/solutions
- Made from LLDPE
- Zero slit accumulation at the bottom
- Meant for residential and industrial purposes

Fig.4.6 Conical Tank

- **Specification of Conical Tank:**

Company	ARK
Body Material	Stainless Steel
Height	600mm
Top Diameter	400mm
Bottom Diameter	25mm

Table 4.6 Specifications of Conical tank

4.7 RESERVOIR TANK :

A Reservoir tank is a container for storing water. Reservoir tanks are used to provide storage of water for use in many applications, drinking water, irrigation agriculture, fire suppression, agricultural farming, both for plants and livestock, chemical manufacturing, food preparation as well as many other uses. Reservoir tank prevents from cavitation to occur and thus avoid problems.

Fig.4.7 Reservoir Tank

➢ Specification:

Company	ARK
Body Material	Stainless Steel
Length	1200 mm
Width	300 mm
Height	350 mm

Table 4.7 Specifications of Reservoir Tank

4.8 CONCLUSION:

All the Hardware components specified in this Chapter are combined to together based on their uses to develop conical tank system .The objective of conical tank system is to control the level of the liquid.

CHAPTER 5

DESIGN AND IMPLEMENTATION

5.1 Introduction:

Today in many industries such as waste water treatment industries, food industries, chemical process plants, petroleum refineries, and pharmaceutical industries, they use control methods for controlling a non-linear conical tank system. Although PID technique have been developed in the Mod-Robs lab , the major drawback faced with PID controller is, it requires large number of parameters to be tuned and takes longer duration to reach settling time . To overcome this drawback, a robust & smart controller is designed. This smart controller not only eliminates requirements of any assumptions, linearization, and modelling of system parameters during controlling action, but also gives accurate and faster settling time results. Reinforcement learning toolkit acts as smart controller in MATLAB Simulink which is being extensively used in robotic applications, as it is less complex and doesn't require the modelling of the system.

5.2 Simulink Block Diagram:

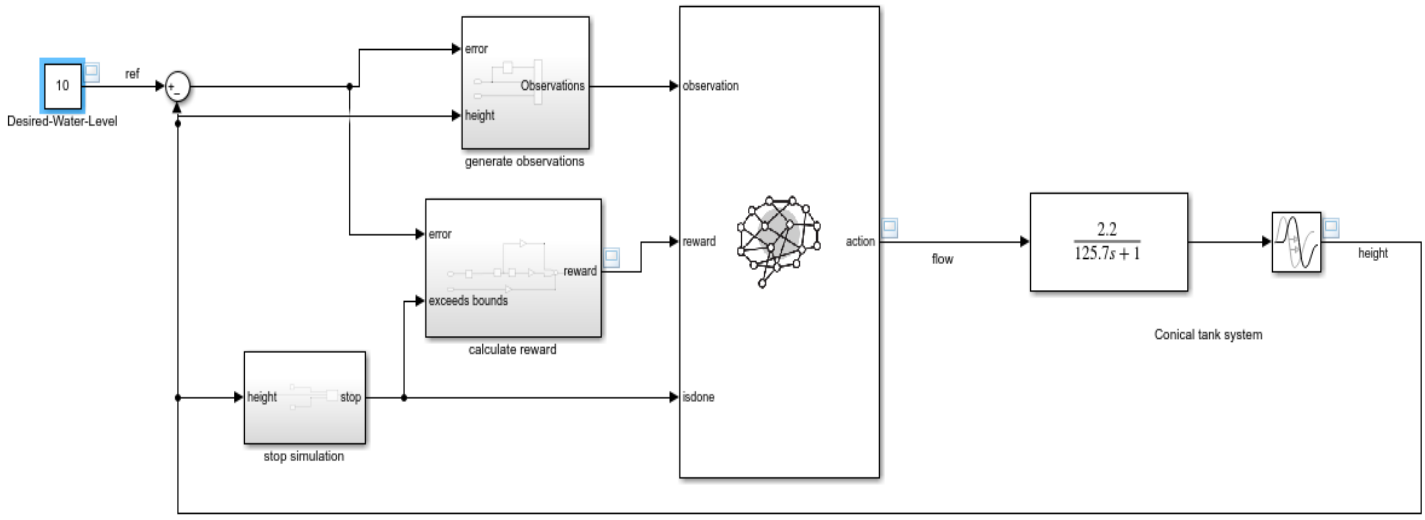


Fig 5.2 Block Diagram

The conical tank system used, is represented in terms of first order transfer function ,which is a mathematical model and the problem is solved using Deep Deterministic Policy Gradient(DDPG) algorithm. The aim is to control the level of liquid in the tank. The agent in this environment is the liquid used, which travels from one state to another. The agent in block diagram takes three inputs as Observation, reward, isdone and gives

output action. The states are different levels or heights of the tank. The maximum height of the experimental tank is 70 cm, which is mapped into 70 states (1 to 71) with intervals of 1 cm each. The goal state would be the desired state (or the set point). The control action is taken in terms of regulating the inflow which is monitored using a rotameter. The range of inflow is from 0 LPH (1 litre per hour = 0.00000028 cubic meters per second) to 350 LPH, mapped into 35 actions (as an example, 1 st action ranges from 0-10 LPH). Thus, for each state, 35 actions are available. The immediate reward is the negative of error i.e. difference between the current state and the set point. The discount factor is kept at 0.99, which makes the future an important factor for the critic value. We have divided the height of the tank into 70 states to make our set point more accurate. But to make the learning easier we can reduce the number of steps. Increasing the number of states and actions, increases the accuracy, but also makes the learning process slow.

5.3 Algorithm Block Representation:

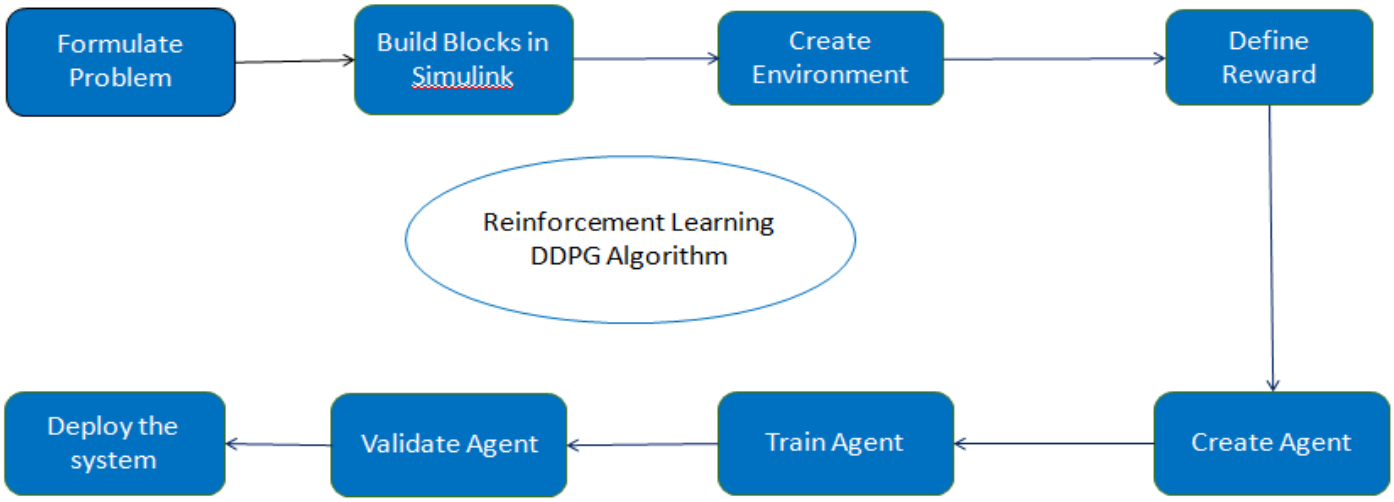


Fig 5.3.1 Algorithm Block Representation

- **Formulate the problem:**

  Conical tank is a classical non-linear control problem whose objective is to control the level of the liquid in the tank. The liquid level in such tanks is a challenging task because of its non-linearity and the continuously changing area of cross section. the major drawback faced with PID controller is, it requires large number of parameters to be tuned and takes longer duration to reach settling time

- **Building Blocks in Simulink:**

  1. Open Matlab Simulink.
     - Syntax:

       open_system('rlconicaltank1')

2. Insert Reinforcement learning Toolkit from the library and give the agent name as ‘Agent’. The RL toolkit takes three input which are: observation, reward, isdone and gives output as action.
3. As per inputs, Connect the observation vector containing integrated error, height & error in one dimension vector, where where h is the height of the tank, e = r − h, and r is the reference height. This gives the present state which the agent is present .

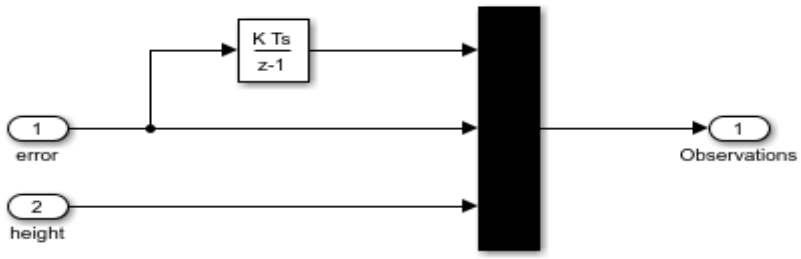


Fig 5.3.2 Observation Internal Block

4. Another input is reward, which is nothing but the error. Set up the reward by reward equation = 10(e < 0. 1) – 1(e ≥ 0 . 1)− 100 (h ≤ 0 h ≥ 70) . This gives the agent an idea about how much is the error caused by the action taken from the previous state.

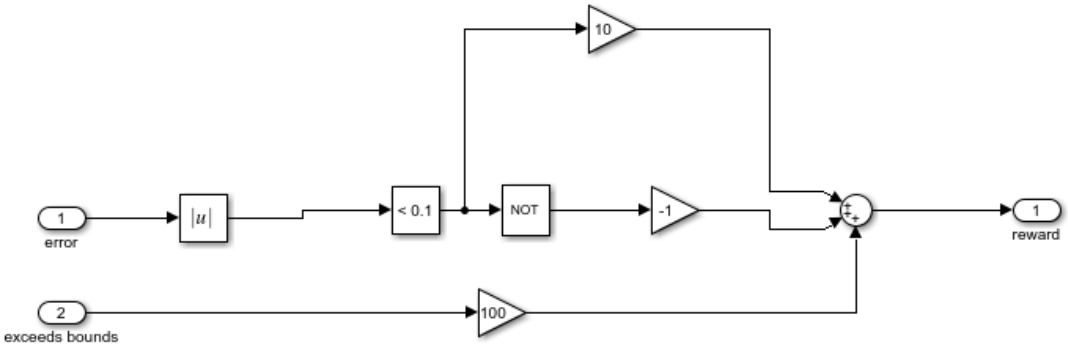


Fig 5.3.3 Reward Internal Block

5. Configure the termination signal such that the simulation stops if h ≤ 0 or h ≥ 70).

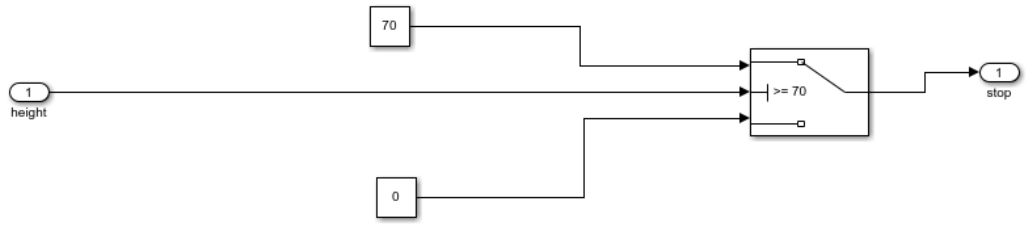


Fig 5.3.4 Stop Simulation Internal Block

➢ Create Environment:

Environment is the Agent's world in which it lives and interacts. The agent can interact with the environment by performing some action but cannot influence the rules or dynamics of the environment by those actions. This means if humans were to be the agent in the earth's environments then we are confined with the laws of physics of the planet. We can interact with the environment with our actions but cannot change the physics of our planet.

When an agent performs an action in the environment, the environment returns a new state of the environment making the agent move to this new state. The environment also sends a Reward to the agent which is a scalar value that acts as feedback for the agent whether its action was good or bad. Creating an environment model includes defining the following: Action and observation signals that the agent uses to interact with the environment. Reward signal that the agent uses to measure its success. Collectively the whole system given as an input to RL toolkit is the environment. Build the environment interface object.

- **Syntax:**

  env = rlSimulinkEnv('rlconicaltank1','rlconicaltank1'/RL Agent',... obsInfo,actInfo);

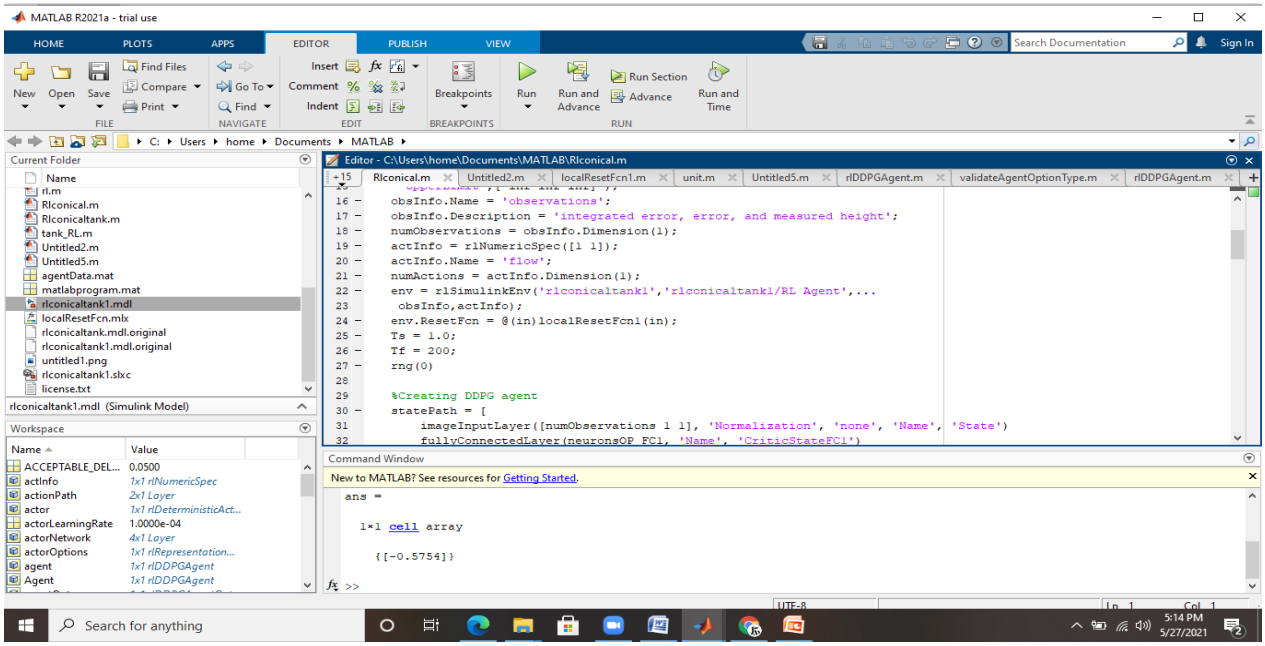


Fig 5.3.5 Environment Execution

➢ Create Agent:

You can create a DDPG agent with default actor and critic representations based on the observation and action specifications from the environment. To do so, perform the following steps.

1. Create observation specifications for your environment. If you already have an environment interface object, you can obtain these specifications using `getObservationInfo`.
2. Create action specifications for your environment. If you already have an environment interface object, you can obtain these specifications using `getActionInfo`.
3. If needed, specify the number of neurons in each learnable layer or whether to use an LSTM layer. To do so, create an agent initialization option object using `rlAgentInitializationOptions`.
4. If needed, specify agent options using an `rlDDPGAgentOptions` object.
5. Create the agent using an `rlDDPGAgent` object.

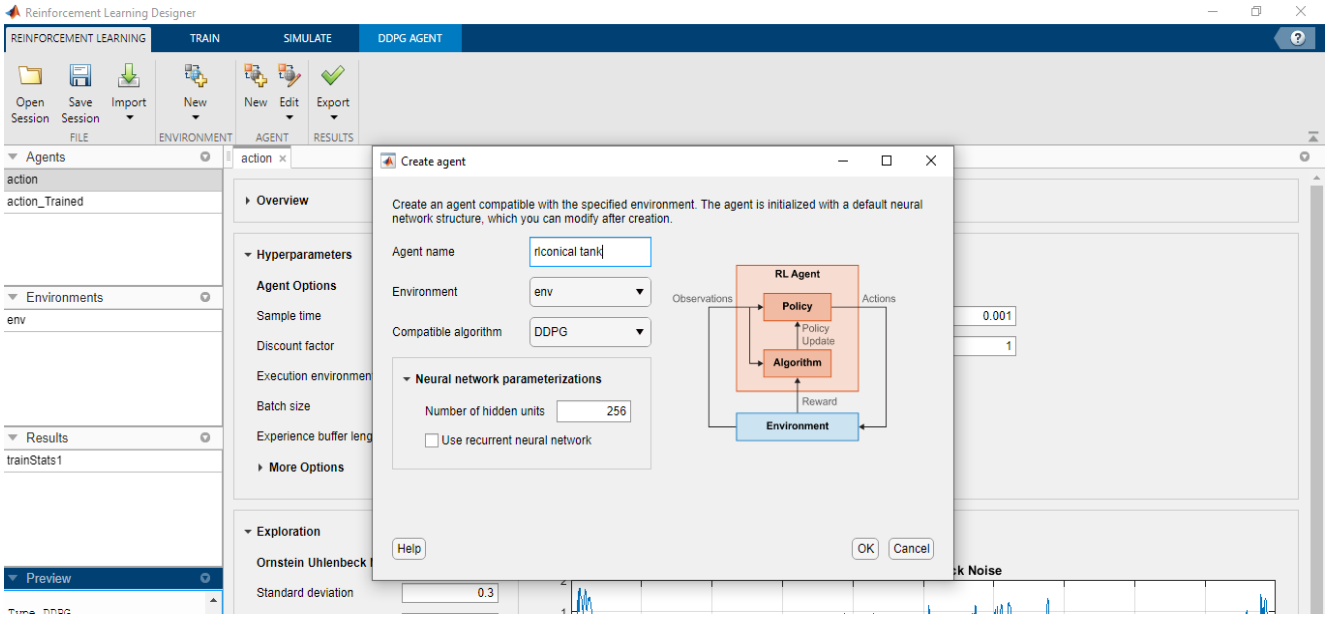


Fig 5.3.6 Agent Creation

➢ Train Agent:

DDPG agents use the following training algorithm, in which they update their actor and critic models at each time step. To configure the training algorithm, specify options using an `rlDDPGAgentOptions` object.

- Initialize the critic $Q(S,A)$ with random parameter values θ_Q, and initialize the target critic with the same random parameter values: $\theta_{Q'}=\theta_Q$.
- Initialize the actor $\mu(S)$ with random parameter values θ_μ, and initialize the target actor with the same parameter values: $\theta_{\mu'}=\theta_\mu$.
- For each training time step:

1. For the current observation S, select action $A = \mu(S) + N$, where N is stochastic noise from the noise model. To configure the noise model, use the `NoiseOptions` option.
2. Execute action A. Observe the reward R and next observation S'.
3. Store the experience (S,A,R,S') in the experience buffer.
4. Sample a random mini-batch of M experiences (S_i,A_i,R_i,S'_i) from the experience buffer. To specify M, use the `MiniBatchSize` option.
5. If S'_i is a terminal state, set the value function target y_i to R_i. Otherwise, set it to

   $y_i=R_i+\gamma Q'(S_i',\mu'(S_i'*\theta_\mu)*\theta_{Q'})$

   The value function target is the sum of the experience reward R_i and the discounted future reward. To specify the discount factor γ, use the `DiscountFactor` option.

   To compute the cumulative reward, the agent first computes a next action by passing the next observation S_i' from the sampled experience to the target actor. The agent finds the cumulative reward by passing the next action to the target critic.
6. Update the critic parameters by minimizing the loss L across all sampled experiences.

$$L = \frac{1}{M}\sum_{i=1}^{M}(y_i - Q(S_i, A_i|\theta_Q))^2$$

7. Update the actor parameters using the following sampled policy gradient to maximize the expected discounted reward.

$$\nabla_{\theta_\mu} J \approx \frac{1}{M}\sum_{i=1}^{M} G_{ai} G_{\mu i}$$

$$G_{ai} = \nabla_A Q(S_i, A|\theta_Q) \quad \text{where } A = \mu(S_i|\theta_\mu)$$

$$G_{\mu i} = \nabla_{\theta_\mu} \mu(S_i|\theta_\mu)$$

Here, G_{ai} is the gradient of the critic output with respect to the action computed by the actor network, and $G_{\mu i}$ is the gradient of the actor output with respect to the actor parameters. Both gradients are evaluated for observation S_i.

8. Update the target actor and critic parameters depending on the target update method.

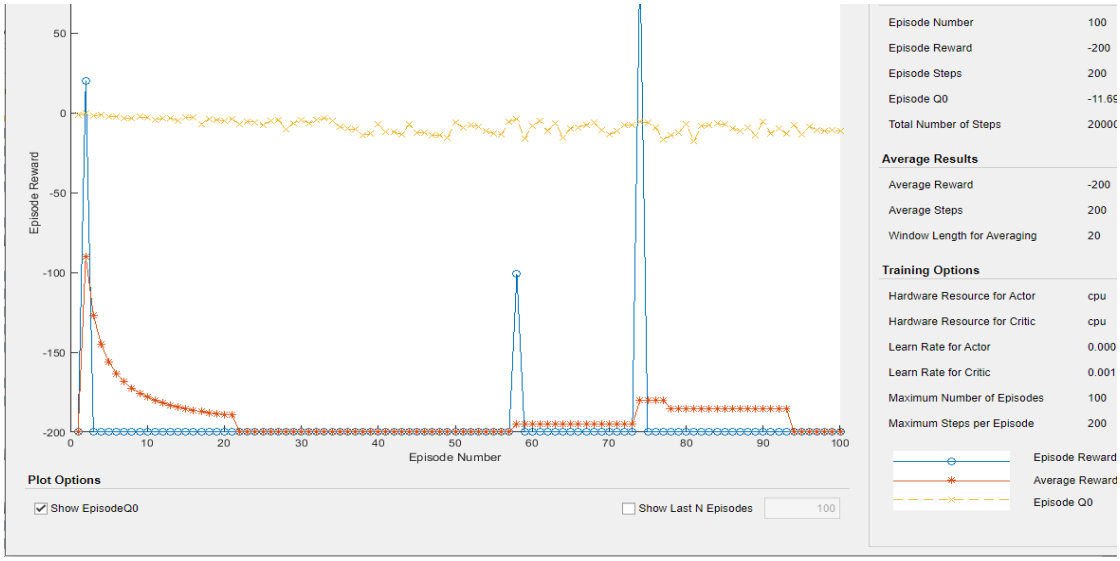


Fig 5.3.7 Training Agent

Reinforcement Learning	**Control System**
Policy	Controller
Environment	The environment includes the plant, the reference signal, and the calculation of the error
Observation	Any measurable value from the environment that is visible to the agent

Action	Manipulated variables or control actions
Reward	Function of the measurement, error signal, or some other performance metric — For example, you can implement reward functions that minimize the steady-state error while minimizing control effort.
Learning Algorithm	Controller liquid level without any tuning of modelling parameters

Table 5.3 DDPG Specifications

- **Validate the Agent:**

Validate the learned agent against the model by simulation.

```
simOpts = rlSimulationOptions('MaxSteps',maxsteps,'StopOnError','on');
experiences = sim(env,agent,simOpts);
```

5.4 CONCLUSION:

The conical tank system problem is solved using Deep Deterministic Policy Gradient(DDPG) algorithm. The aim is to control the level of liquid in the tank. Environment is created after building blocks in the Simulink as per the algorithm. The agent in this environment is the liquid used, which travels from one state to another. The learning process starts with the initial brain of a controller which is the Actor Critic Matrix . The learning is divided into several trials. If the liquid reaches the goal state, trial is considered a success, and the next trial is started. After each trial the Critic is updated, and carried over to the next trial. Basically, learning is continued in all the trials. With each trial efficiency of the system improves and level settles faster. After enough trials, when there is no further change in settling times, the learning is stopped. Then the action is executed by sending the flow rate information to the pneumatic valve, and thus liquid moves to the next level which is the next state. The current state is updated to the new state, and again the best possible action is obtained. This process of updating the Critic is repeated until the level reaches the desired state.

CHAPTER 6

RESULT

➢ Path of Agent:

Actor network takes the observation and gives the action, followed by critic network which takes observation and corresponding action as inputs & give Q value as output, Q value which is a reward measures how good the action is. (Q value = reward+ discount value*Q(next)). The maximizing and minimizing Q value gives the actor network to train and reach desired set point. The next observation(state) and actor output given as an input to critic network which gives Q(next) as output . To implement target network DDPG agent follows the path which makes Q equal to desired Q network.

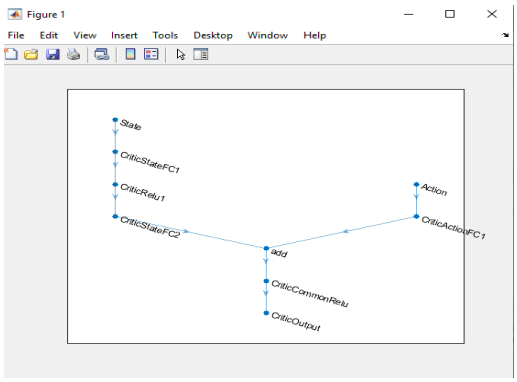


Fig 6.1 Agent Path

➢ Simulation results:

The model is trained in the MATLAB IDE and trained model is deployed in the RL toolkit. The maximum height of the experimental tank is 70 cm, which is mapped into 70 states (1 to 71) with intervals of 1 cm each. The goal state would be the desired state (or the set point 9 cm). The control action is taken in terms of regulating the inflow which is monitored using a rotameter. The range of inflow is from 0 LPH (1 litre per hour = 0.00000028 cubic meters per second) to 350 LPH, mapped into 35 actions (as an example, 1st action ranges from 0-10 LPH). The agent in the Simulink environment takes three inputs Observation (states), Reward, Is done, the output of the RL toolkit is the action which is obtained by actor network. The actor output is shown in Fig 6.3, If set point is not reached then the action output is given as an input to critic network which is nothing but reward signal. The reward output is shown in Fig 6.4. Finally, based on this input parameters deployed in local reset function code, the algorithm runs and set point (height) is reached faster, without any peak over shoot, faster rise time and faster settling time. The height output is shown in the Fig 6.2. Different set points have been given to the model and output is verified. For set point 8cm,2cm (Fig 6.5, Fig 6.6) we have recognized a peak overshoot which is occurred as the model have been testing its trial and error method with the critic network, This is can be overcome by giving large combination of training parameters in the local reset function.

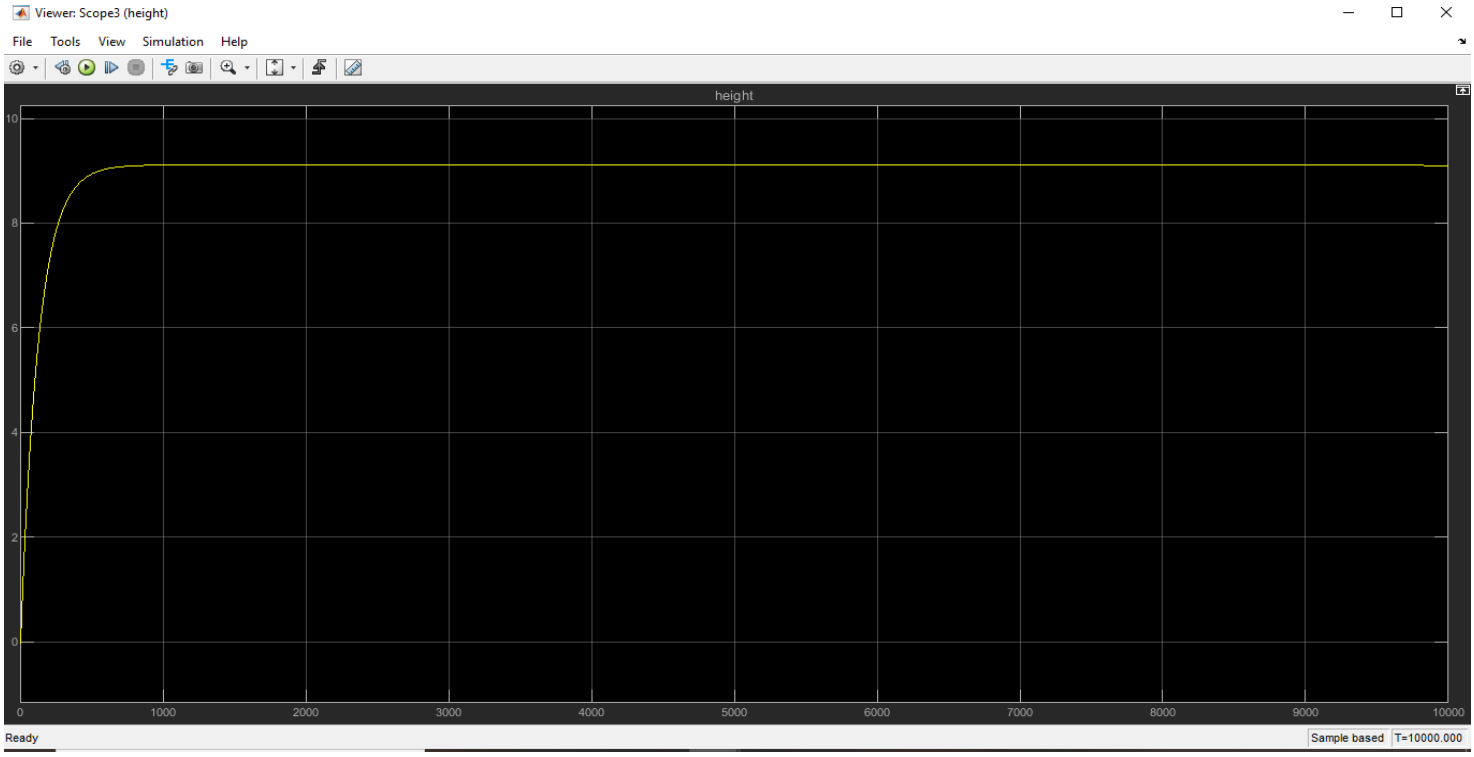


Fig 6.2 Height output

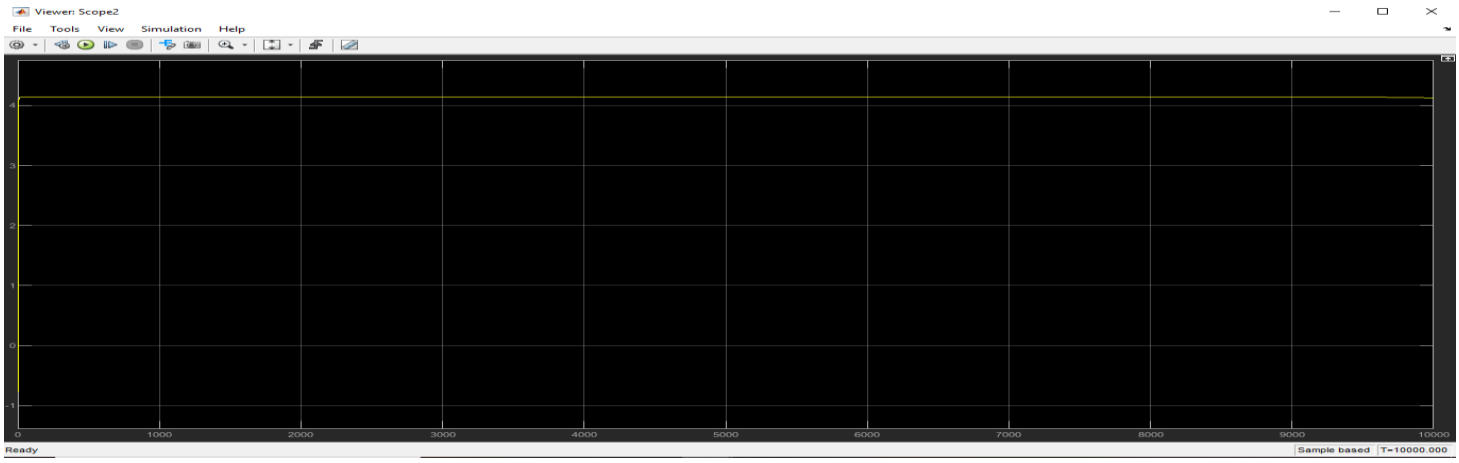


Fig 6.3 Action Output

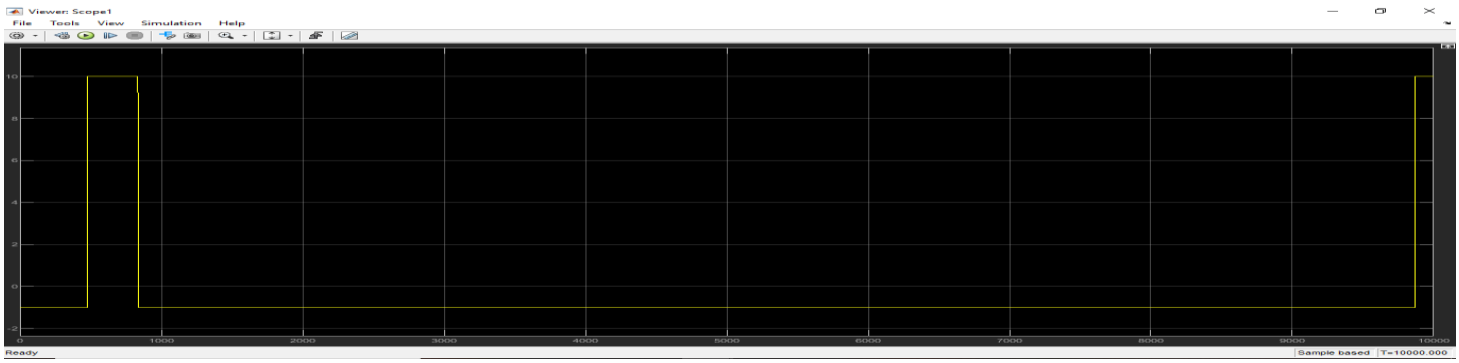


Fig 6.4 Reward Output

- **Set point 8:**

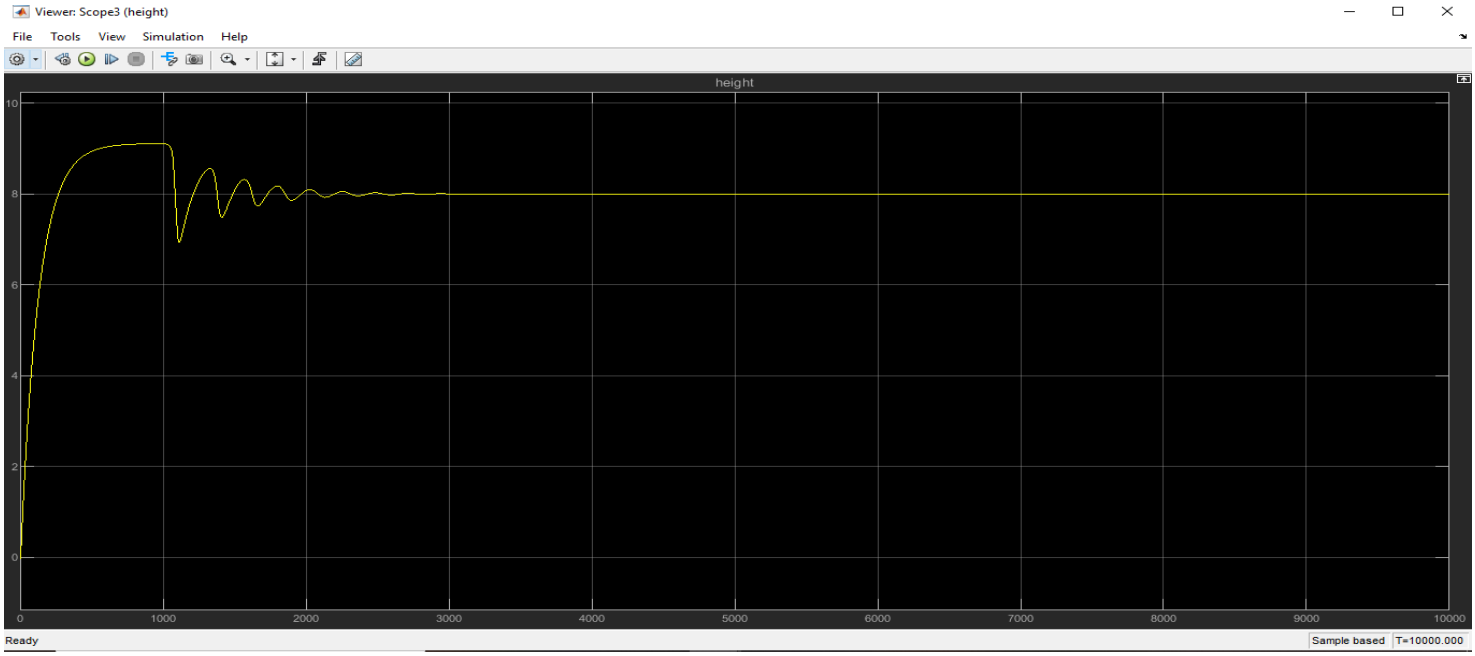


Fig 6.5 Height output with Set point 8

- **Set point 2:**

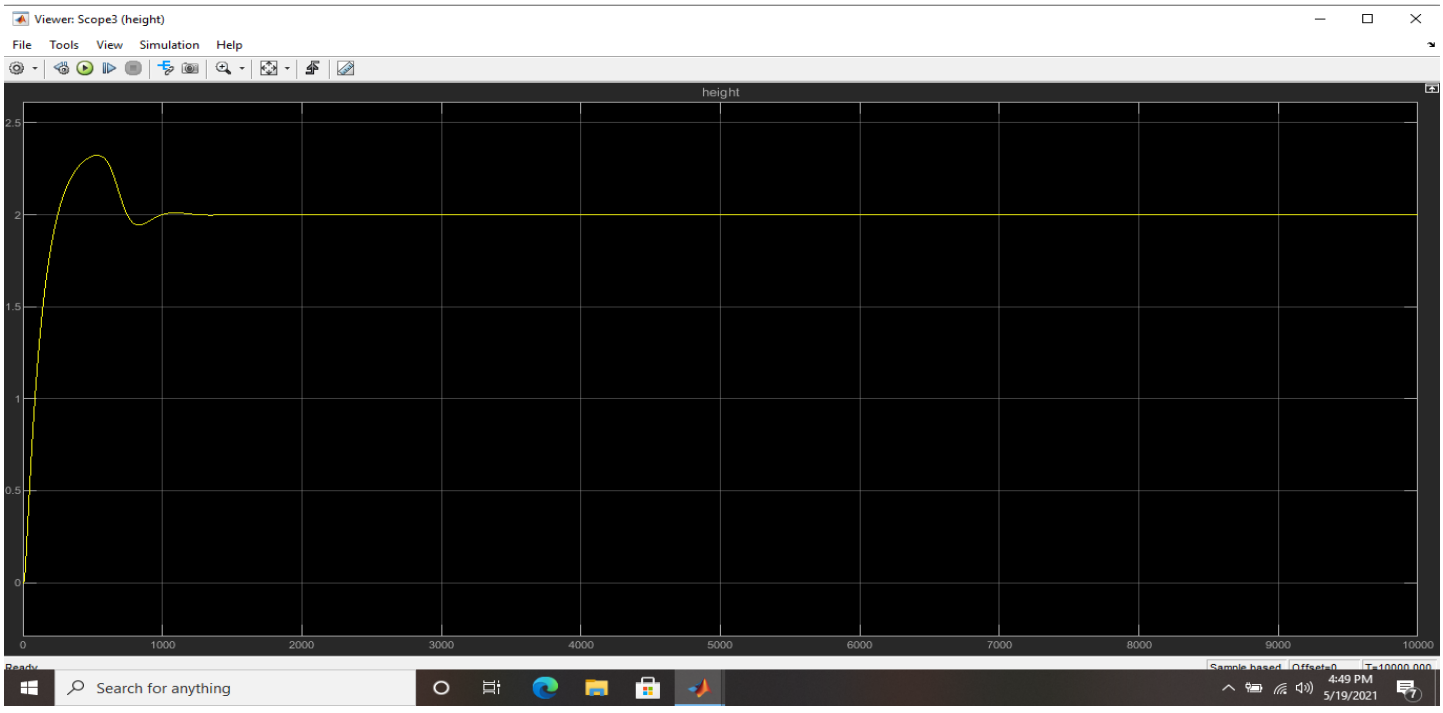


Fig 6.6 Height Output with Set point 2

CHAPTER 7

CONCLUSION

A Machine learning based controller is designed for the controlling the level of liquid in conical tank system using Deep Deterministic Policy Gradient algorithm, a type of reinforcement learning algorithm. First the path of the agent is obtained by running simulation in MATLAB, without interfacing directly with the mathematical modeling system. In the MATLAB Simulink, from library many different blocks mainly including reinforcement learning toolkit is deployed and interfaced with the DDPG algorithm code in MATLAB. Later through Actor critic operations the state in the form of height of the liquid is updated by many trials in the experimentation until it reached desired water level, and then implemented. This method prevented the system from damages due to flow disturbances, and rejected unknown disturbances. From the Figure 6.2, 6.5, 6.6 with different set points, it is clearly inferred that the level of the tank settled at the set point. Figure 5.2 is an example of the robustness of the designed controller, Fig 6.5 shows that even after sudden random disturbances the controller controls the inflow of the liquid, and the liquid settled at the desired level. Thus, a robust and efficient controller based on reinforcement learning algorithm is designed for a non-linear conical tank process .The proposed controller based on machine learning does not require prior information of the system as compared to PID and fuzzy techniques. It eliminates the use of designing the system using differential equations, and finding the system transfer functions. The designed controller can be easily used with other non-linear systems with basic changes in states and actions.

CHAPTER 8

SCOPE FOR FUTURE WORK

During the learning process, the output valve which controls the outflow of liquid from the tank was made constant. Also, it should be noted that only the linear portions of the inflow rate were taken as a mathematical model while deciding the number of actions for faster learning. The controller can be made to learn with different positions of the output valve and all possible actions for better and robust control. Thus, different rates of inflow and outflow can be integrated in the learning process. This algorithm can be equipped as an input to many non-linear systems but changes must be made in the local reset function in the MATLAB code on the parameter to be measured. In reinforcement learning algorithm, the training of the model is very slow but once if the model is trained and deployed in the reinforcement learning toolkit, then output response is obtained very fast and accurate. Researchers are working really hard to make training of the model faster, that eliminates waiting time while the agent is training. Once this is done then Machine learning is going to triumph in the domain of controller design.

BIBBLIOGRAPHY

Textbooks

1. PID Controllers: Theory, Design and Tuning , Karl J. Astrom (Author), T. Hagglund (Author),1995
2. Intelligent Control Design and MATLAB, Jinkin Liu,2018
3. Intelligent Control, Das Sharma,2018
4. Reinforcement Learning and Optimal control book, Athena Scientific, July 2019
5. Robotics, Vision & Contol,Corke,2017
6. Machine Learning, Tom m. Mitchell, 2013.
7. Reinforcement learning User guide, Mathworks R2021a,2020.

REFERENCES

1. Bhuvaneswari, N. S., Uma, G., & Rangaswamy, T. R. (2009). Adaptive and optimal control of a non-linear process using intelligent controllers. Applied Soft Computing, 9(1), 182-190.
2. Pandimadevi, & Ganesan, Pandimadevi & Kumar, Dr.V.Selva. (2018). DESIGN OF PI CONTROLLER FOR A CONICAL TANK SYSTEM. World Journal of Engineering and Technology. 4. 165-175.
3. Aravind, P., & Kumar, S. G. (2013). Optimal tuning of PI controller using swarm intelligence for a nonlinear process. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2(12), 5949-5958.
4. Nivetha, J. & Velappan, Vijayan. (2016). Design of tunable method of PID controller for conical tank system. 251-254. 10.1109/ICCPEIC.2016.7557204.
5. Rupinder Jawanda, 2014, Performance Analysis of Intelligent Controller and Classical PID Controller for Conical Tank System, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 03, Issue 07 (July 2014)
6. Saravanakumar, G. and Dinesh, S. and Preteep, S. and Sridhar, P. and M, suresh, Controller Tuning Method for Non-Linear Conical Tank System (March 11, 2017). Asian Journal of Applied Science and Technology (AJAST), Volume 1, Issue 2, Pages 224-228, March 2017,
7. P. Aravind, M. Valluvan, S. Ranganathan. Modelling and simulation of non linear

tank. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2013, 2(2): 842–849.

8. N. Gireesh, G. Sreenivasulu. Comparison of PI controller performances for a conical tank process using different tuning methods. International Conference on Advances In Electrical Engineering, Vellore, India: IEEE, 2014.

9. H. Kala, P. Aravind, M. Valluvan. Comparative analysis of different controller for nonlinear level control process. IEEE Conference on Information and Communication Technologies, South Korea: IEEE, 2013: 724–729.

10. J. Zhong. *PID Controller Tuning: A Short Tutorial*. West Lafayette, U.S.A.: Purdue University, 2006.

11. Anandanatarajan, R., Chidambaram, M., & Jayasingh, T. (2006). Limitations of a PI controller for a first-order nonlinear process with dead time. ISA transactions, 45(2), 185- 199.

12. Alvarez, E., Riverol, C., & Navaza, J. M. (1999). Control of chemical processes using neural networks: implementation in a plant for xylose production. ISA transactions, 38(4), 375-382.

13. Nk, Thara & M M, Shinu & Devadhas, Glan & Mohan, Dhanoj. (2018). Control Schemes For a Nonlinear Conical tank System. 610-613. 10.1109/ICCPCCT.2018.8574267.

14. Madhubala, T. K., Boopathy, M., Chandra, J. S., & Radhakrishnan, T. K. (2004). Development and tuning of fuzzy controller for a conical level system. In Intelligent Sensing and Information Processing, 2004. Proceedings of International Conference on (pp. 450- 455). IEEE.

15. Noel, M. M., & Pandian, B. J. (2014). Control of a nonlinear liquid level system using a new artificial neural network based reinforcement learning approach. Applied Soft Computing, 23, 444-451.

16. Ono, S., Inagaki, Y., Aisu, H., Sugie, H., & Unemi, T. (1995, March). Fast and feasible reinforcement learning algorithm. In Fuzzy Systems, 1995. International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium., Proceedings of 1995 IEEE Int (Vol. 3, pp. 1713-1718). IEEE.

17. Boubertakh, H., Tadjine, M., Glorennec, P. Y., & Labiod, S. (2010). Tuning fuzzy PD and PI controllers using reinforcement learning. ISA transactions, 49(4), 543-551.

18. Yen, G., & Hickey, T. (2002). Reinforcement learning algorithms for robotic navigation in dynamic environments. In Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on (Vol. 2, pp. 1444-1449). IEEE.

19. Mohajerin, N., Menhaj, M. B., & Doustmohammadi, A. (2010, July). A reinforcement learning fuzzy controller for the ball and plate system. In Fuzzy Systems (FUZZ), 2010 IEEE International Conference on (pp. 1-8). IEEE.

20. Anfu, Guo & Zhang, & Zheng, & Yanhua, Du. (2020). An Autonomous Path Planning Model for Unmanned Ships Based on Deep Reinforcement Learning. Sensors. 20. 426. 10.3390/s20020426.

21. Guo, M., Liu, Y., & Malec, J. (2004). A new Q-learning algorithm based on the metropolis criterion. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 34(5), 2140-2143.

22. Prabhu Ramanathan, Kamal Kant Mangla, Sitanshu Satpathy, Smart controller for conical tank system using reinforcement learning algorithm, Measurement,Volume 116,2018,Pages 422-428,ISSN 0263-2241,

23. Mehta, Deepanshu. (2020). State-of-the-Art Reinforcement Learning Algorithms. International Journal of Engineering and Technical Research. V8. 10.17577/IJERTV8IS120332.

24. Dankwa, Stephen & Zheng, Wenfeng. (2019). Twin-Delayed DDPG: A Deep Reinforcement Learning Technique to Model a Continuous Movement of an Intelligent Robot Agent. 1-5. 10.1145/3387168.3387199.

APPENDIX

➢ PROGRAM CODE:

```
TANK_SIMULATION_MODEL = 'rlconicaltank1';
 MAX_EPISODES = 100;
AVERAGE_WINDOW = 50;        % Average over 50 time-steps
ACCEPTABLE_DELTA = 0.05;
% DDPG Hyper-paramaters
criticLearningRate = 1e-03;
actorLearningRate  = 1e-04;
GAMMA = 0.90;
BATCH_SIZE = 64;
% Critic network: Neurons for fully-connected Observsation (state) and Action paths
neuronsOP_FC1 = 50; neuronsOP_FC2 = 25; neuronsAP_FC1 = 25;

obsInfo = rlNumericSpec([3 1],...
 'LowerLimit',[-inf -inf 0 ]',...
 'UpperLimit',[ inf inf inf]');
obsInfo.Name = 'observations';
obsInfo.Description = 'integrated error, error, and measured height';
numObservations = obsInfo.Dimension(1);
actInfo = rlNumericSpec([1 1]);
actInfo.Name = 'flow';
numActions = actInfo.Dimension(1);
env = rlSimulinkEnv('rlconicaltank1','rlconicaltank1/RL Agent',...
 obsInfo,actInfo);
env.ResetFcn = @(in)localResetFcn1(in);
Ts = 1.0;
Tf = 200;
rng(0)

%Creating DDPG agent
```

```
statePath = [
    imageInputLayer([numObservations 1 1], 'Normalization', 'none', 'Name', 'State')
    fullyConnectedLayer(neuronsOP_FC1, 'Name', 'CriticStateFC1')
    reluLayer('Name', 'CriticRelu1')
    fullyConnectedLayer(neuronsOP_FC2, 'Name', 'CriticStateFC2')];
actionPath = [
    imageInputLayer([numActions 1 1], 'Normalization', 'none', 'Name', 'Action')
fullyConnectedLayer(neuronsAP_FC1, 'Name', 'CriticActionFC1')];
commonPath = [
additionLayer(2,'Name', 'add')
reluLayer('Name','CriticCommonRelu')
fullyConnectedLayer(1, 'Name', 'CriticOutput')];

criticNetwork = layerGraph();
criticNetwork = addLayers(criticNetwork,statePath);
criticNetwork = addLayers(criticNetwork,actionPath);
criticNetwork = addLayers(criticNetwork,commonPath);
criticNetwork = connectLayers(criticNetwork,'CriticStateFC2','add/in1');
criticNetwork = connectLayers(criticNetwork,'CriticActionFC1','add/in2');
figure
plot(criticNetwork)
criticOpts = rlRepresentationOptions('LearnRate', criticLearningRate, 'GradientThreshold',1);
critic = rlRepresentation(criticNetwork,obsInfo,actInfo,'Observation',...
    {'State'},'Action',{'Action'},criticOpts);
actorNetwork = [
    imageInputLayer([numObservations 1 1], 'Normalization', 'none', 'Name', 'State')
    fullyConnectedLayer(3, 'Name', 'actorFC')
    tanhLayer('Name', 'actorTanh')
    fullyConnectedLayer(numActions, 'Name', 'Action')
    ];
actorOptions = rlRepresentationOptions('LearnRate',actorLearningRate,'GradientThreshold',1);
%actor =
rlRepresentation(actorNetwork,obsInfo,actInfo,'Observation',{'State'},'Action',{'Action'},actorOptions);
```

```
actor = rlDeterministicActorRepresentation(actorNetwork,obsInfo,actInfo,'Observation',{

    'State'},'Action',{

    'Action'},actorOptions);
agentOpts = rlDDPGAgentOptions(...
    'SampleTime', Ts,...
    'TargetSmoothFactor', 1e-3,...
    'DiscountFactor', GAMMA, ...
    'MiniBatchSize', BATCH_SIZE, ...
    'ExperienceBufferLength', 1e6, ...
    'ResetExperienceBufferBeforeTraining', false, ...
    'SaveExperienceBufferWithAgent', false);
  initOpts = rlAgentInitializationOptions('NumHiddenUnit',64,'UseRNN',true);

  % Exploration Parameters
  % ----------------------
  % Ornstein Uhlenbeck (OU) action noise:
  %   Variance*sqrt(SampleTime) keep between 1% and 10% of your action range
  % Variance is halved by these many samples:
  %       halflife = log(0.5)/log(1-VarianceDecayRate)
  % Default: 0.45, 1e-5
  DDPG_Variance = 1.5;
  DDPG_VarianceDecayRate = 1e-5; % Half-life of 1,000 episodes
  % agentOpts.NoiseOptions.Variance = DDPG_Variance;
  % agentOpts.NoiseOptions.VarianceDecayRate = DDPG_VarianceDecayRate;
  %action = rlDDPGAgent(obsInfo,actInfo,initOpts);
  % load predefined environment

  % obtain observation and action specifications
  obsInfo = getObservationInfo(env);
  actInfo = getActionInfo(env);
  ts = 0.1 ; % sample time
  t = 0:ts:10; % time vector
```

```
%Observations  = sin(t);% your signal
Agent = rlDDPGAgent(actor,critic,agentOpts);
%Agent = rlDDPGAgent(10,10,3);
getAction(Agent,{rand(obsInfo(1).Dimension)})
generatePolicyFunction(Agent)
maxepisodes = MAX_EPISODES;
maxsteps = ceil(Tf/Ts);
% For parallel computing: 'UseParallel',true, ...
%criticOptions.UseDevice = 'gpu';
%agent = rlDDPGAgent(actor,critic,agentOpts);
%maxepisodes = 5000;
%maxsteps = ceil(Tf/Ts);
trainOpts = rlTrainingOptions(...
 'MaxEpisodes',maxepisodes, ...
 'MaxStepsPerEpisode',maxsteps, ...
 'ScoreAveragingWindowLength',20, ...
 'Verbose',false, ...
 'Plots','training-progress',...
 'StopTrainingCriteria','AverageReward',...
 'StopTrainingValue',800);
% Load the pretrained agent for the example.
trainingresults = train(Agent,env,trainOpts);
load('WaterTankDDPG.mat','agent')

rng(0);
simOptions = rlSimulationOptions('MaxSteps',200,'NumSimulations',1);
experience = sim(env,Agent,simOptions);
```

“local reset function:”

```
function in = localResetFcn1(in)
% Randomize reference signal
blk = sprintf('rlconicaltank1/Desired-Water-Level');
```

```
heightValues =
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,
39,40,41,42,43,44,45,46,47,48,49,50,52,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70];
idx = randperm(length(heightValues),1);
height = heightValues(idx);
%function in = localResetFcn(in)
% randomize reference signal
%blk = sprintf('rlwatertank/Desired \nWater Level');
in = setBlockParameter(in,blk,'Value',num2str(height));
% randomize initial height
%h = 3*randn + 10;
%%while h <= 0 || h >= 20
 %h = 3*randn + 10;
%end
%blk = 'rlconicaltank1/Water-Tank System/H';
%in = setBlockParameter(in,blk,'InitialCondition',num2str(h));
%end

%height = 3*randn + 10;
%global state_desired
%Reward = -abs(state(1)-state_desired);

%while height <= 0 || height >= 20
   %height = 3*randn + 10;
%end
%in = setBlockParameter(in,blk,'Value',num2str(height));
%'flow',[-5 0 5],'rlconicaltank'
% Randomize initial height
%height = 3* randn + 10;
%while height <= 0 || height >= 20
   %height = 3*randn + 10;
%end
%blk = 'rlconicaltank1/Desired-Water-Level/height';
%in = setBlockParameter(in,blk,'InitialCondition',num2str(height));
```

```
end
```

‘rlDDPG Agent’

```
narginchk(2,4)
AgentType = "DDPG";
FirstArg = varargin{1};
if isa(FirstArg,'rl.representation.rlDeterministicActorRepresentation')
    % AGENT = RLDDPGAGENT(ACTOR,CRITIC)
    % AGENT = RLDDPGAGENT(ACTOR,CRITIC,AGENTOPTIONS)
    Actor = FirstArg;
    Critic = varargin{2};
    if nargin < 3
        AgentOptions = rl.util.getDefaultAgentOptions(AgentType,hasState(Actor));
    else
        AgentOptions = varargin{3};
    end

elseif isa(FirstArg,'rl.util.RLDataSpec')
    % AGENT = rlDDPGAgent(OINFO,AINFO)
    % AGENT = rlDDPGAgent(OINFO,AINFO,INITOPTIONS)
    % AGENT = rlDDPGAgent(OINFO,AINFO,AGENTOPTIONS)
    % AGENT = rlDDPGAgent(OINFO,AINFO,INITOPTIONS,AGENTOPTIONS)
    [ObservationInfo,ActionInfo,InitOptions,AgentOptions] =
rl.util.parseAgentInitializationInputs(AgentType,varargin{:});
    Actor = rl.representation.rlDeterministicActorRepresentation.createDefault(Observatinfo, ActionInfo,
InitOptions);
    Critic = rl.representation.rlQValueRepresentation.createDefault(ObservationInfo, ActionInfo, InitOptions,
'singleOutput');
else
    error(message('rl:agent:errDPGInvalidFirstArg'));
end
```

```
rl.util.validateAgentOptionType(AgentOptions,AgentType);
Agent = rl.agent.rlDDPGAgent(Actor, Critic, AgentOptions);
```