

APPENDIX A

SOFTWARE DEVELOPMENT

List of software and developed files for multiple model adaptive controllers are as below.

A1:List Software used to develop the algorithm

Version of software	Description	Company
MATLABVersion R2013a, 8.1, 32-bit (win 32)	For simulation & overall development of whole algorithm	Mathwork, USA www.mathwork.com
CCS Platinum v3.3 32-bit (win 32)	For interface real time system & MATLAB	Texas Instrument www.ti.com

A2:Graphical user interface development (GUI)

Name of files	Description
MAIN_GUI.fig	Main screen of GUI
COMPARISION.fig	Comparisons of all developed controller
MMAC.fig	GUI for Multiple model adaptive controller
OPTIMIZATIO_GA.fig	Optimization using Genetic algorithm
OPTIMIZATIO_PSO.fig	Optimization using particle swarm optimization
Speed.fig	GUI shows the speed calculation
angle.fig	GUI shows the Rotor angle calculation
torque.fig	GUI shows the torque calculation
Dwm1.fig	GUI for change in speed of m/c 1 & 2 for MMAC
linepower.fig	GUI for line power of multimachine system for MMAC
terminalvoltage.fig	GUI for terminal voltage of multimachine system for MMAC

A3:Simulink files

Name of files	Description
Without_PSS.slx	Heffron Philips model without PSS for SMIB system
CPSScase1.slx	Conventional power system stabilizer for light loading condition
CPSScase2.slx	Conventional power system stabilizer for normal loading condition
CPSScase3.slx	Conventional power system stabilizer for heavy loading condition
DualPSScase1.slx	Dual i/p PSS for light loading condition
DualPSScase2.slx	Dual i/p PSS for normal loading condition
DualPSScase3.slx	Dual i/p PSS for heavy loading condition
FPSScase1.slx	Fuzzy PSS for light loading condition
FPSScase2.slx	Fuzzy PSS for normal loading condition
FPSScase3.slx	Fuzzy PSS for heavy loading condition
ANNPSScase1.slx	Artificial Neural network based PSS for light loading condition
ANNPSScase2.slx	Artificial Neural network based PSS for normal loading condition
ANNPSScase3.slx	Artificial Neural network based PSS for heavy loading condition
gacase1.slx	Genetic algorithm based PSS for light loading condition
gacase2.slx	Genetic algorithm based PSS for normal loading condition
gacase3.slx	Genetic algorithm based PSS for heavy loading condition
without_mmc.slx	Multimachine system without PSS
conventional_pss.slx	Multimachine system with PSS
neural_pss.slx	ANN based PSS for Multimachine system
fuzzy_pss.slx	Fuzzy logic based PSS for Multimachine system
ga_pss.slx	Genetic algorithm based PSS for Multimachine system
Only_pss_discrete.slx	Discrete system model for SMIB system
mmc_pss_discrete.slx	Discrete system model for Multimachine system
Pso_smib.slx	Particle swarm optimization model for SMIB system

Pso_mmc.slx	Particle swarm optimization model for Multimachine system
-------------	---

A4:MATLAB files

Name of files	Description
b10to2.m	Conversion of base 10 to base 2
calculateobj.m	Calculate the main objective function for GA
gapgm.m	Main file for GA
decode1.m	Conversion from binary to variable representation
encode1.m	Conversion from variable to binary representation
init.m	Creation of random population
mate.m	Randomly reorders (mates) OLD_GEN
mutate.m	Changes a gene of the OLD_GEN with probability Pm
reproduce.m	selects individuals proportional to their fitness
score3.m	computes the fitness and the objective function values of a population
xover.m	Creates a NEW_GEN from OLD_GEN
nn_con.m	Artificial neural network coding for conventional PSS
nbnkp.m	Artificial neural network coding for conventional PSS using back propagation algorithm
calculate.m	Calculate the main objective function for GA
main_PSO.m	Main file for PSO

A5:FUZZY files

Name of files	Description
RMMAC.fis	Fuzzy file for multiple model adaptive control
RMMAC_A.fis	Fuzzy file for multiple model adaptive controller for Intelligent control
RMMAC_Ss.fis	Fuzzy file for multiple model adaptive controller for Smart control
Psssmib.fis	Fuzzy file for single machine infinite bus system
Multimachinepss.fis	Fuzzy file for multimachine infinite bus system

APPENDIX B

SYSTEM DATA

List of software and developed files for multiple model adaptive controllers are as below.

B1:Single Machine Data

The Generator data:

Xd	1.6
Xq	1.55
x'd	0.32
T'd0	6.0
H	5
F	50 Hz.

The Transmission line data:

Re	0
Xe	0.4

AVR data:

KA	200
TA	0

PSS data:

T1	0.154 sec
T2sec	0.033 sec
TW	1.4 sec
Kstab	9.5

Dual PSS data:

Ks1	-0.5
Ks2	48.25
T1	0.05 sec
T2	0.25 sec
Tw1,Tw2	0.2 s

The calculation of K1 to K6:

$$P_t = \frac{V_t E_b \sin \theta_o}{X_e} \hat{I}_a = \frac{\widehat{V}_t - E_b < 0}{j X_e}$$

$$V_{do} = -V_{to} \sin(\delta_o - \theta_o) \quad V_{qo} = V_{to} \cos(\delta_o - \theta_o)$$

$$\widehat{E}_{qo} = V_t < \theta_o + j x_q \hat{I}_a$$

$$E'_{qo} = E_{fdo} + (X_d - X_d') i_{do}$$

$$i_{do} = -I_a \sin(\delta_o - \phi_o) \quad i_{qo} = I_a \cos(\delta_o - \phi_o)$$

$$E_{fdo} = E_{qo} - (X_d - X_q) i_{do}$$

$$K1 = \frac{E_b E_{qo} \cos \delta_o}{X_e + X_q} + \frac{(X_q - X_d')}{(X_e + X_d')} E_b i_{qo} \sin \delta_o$$

$$K2 = \frac{(X_e + X_q)}{(X_e + X_d')} i_{qo} = \frac{E_b \sin \delta_o}{X_e + X_d'}$$

$$K3 = \frac{X_e + X_d'}{X_d + X_e}$$

$$K4 = \frac{(X_d - X_d')}{(X_d' + X_e)} E_b \sin \delta_o$$

$$K5 = \frac{-X_q V_{do} E_b \cos \delta_o}{(X_e + X_q) V_{to}} - \frac{X_d' V_{qo} E_b \sin \delta_o}{(X_e + X_d') V_{to}}$$

$$K6 = \frac{X_e}{X_e + X_d'}$$

B2:Multimachine Data

Generator-1 Data

Nominal Power (Pn)VA	1000 MVA
Line-line Voltage Vn (rms)	13800
Xd	1.305
Xd'	0.296
Xd''	0.252
Xq	0.474
Xq''	0.243
Xl	0.18
Inertia Constant H	3.7
Friction Factor	0
Pole Pair P	4
Theta	-16.68 (degree)
Rotor Type	Salient Pole

Generator-2 Data

Nominal Power (Pn)VA	500 MVA
Line-line Voltage Vn (rms)	1380
Xd	1.305
Xd'	0.296
Xd''	0.252
Xq	0.474
Xq''	0.243
Xl	0.18
Inertia Constant H	3.7
Friction Factor	0
Pole Pair P	32
Theta	-69.6
Rotor Type	Salient Pole

APPENDIX C

LIST OF RESEARCH PUBLICATIONS

Following is the list of our publications, presentations relevant to the work included in the thesis.

C1:International Journal

- (1) Ami Patel, Prof. S.K.Shah, “Design and Analysis of Switched Multiple Model Adaptive Control for Local Controllers” at International Journal of Engineering Associates, ISSN: 2320-0804, Vol. 1 Issue 4,
2012.<http://www.advanceresearchlibrary.com/temp/downloads/ijea/feb2013/rk32.pdf>)
- (2) Ami Patel, Prof. S.K.Shah, “Development of Real time controller of a Single Machine Infinite Bus system with PSS”, International Journal of Electrical Engineering (IJEE), ISSN 2321-600X, Volume 2, Issue 9, September 2014, Impact Factor: 1.318.
- (3) Ami Patel, Hemisha Patel, “Comparison of Different Design Methods for power System Stabilizer Design - A Review”, International Journal for Scientific Research & Development, ISSN (online): 2321-0613, Vol. 2, Issue 08, 2014.

C2:Peer-reviewed international conferences

- (4) Ami Patel, Prof. S.K.Shah, Hardik A Shah, “Design of fuzzy logic power system stabilizers in a multimachine power system using Particle swarm optimization based optimal control algorithm”, Discovery International Daily journal ISSN 2278 – 5469.
- (5) Ami Patel, Hemisha Patel, Jay S Tandel, “Development of Intelligent controller for Power System stabilization for Single Machine Infinite Bus system”, Discovery International Daily journal ISSN 2278 – 5469.

(6) Ami Patel, Hemisha Patel “Performance Evaluation of PSS Under Different Loading Condition”, Global Conference on communication technologies GCCT, IEEE Conference at Tamilnadu during 23-24 April,2015.

C3:Conference Proceeding

(7) Ami Patel, Prof. S.K.Shah, Hardik A Shah “Improvement of Transient stability of SMIB system using Fuzzy & ANFIS based STATCOM damping stabilizer”, Target -2014, Institution of electrical and electronics engineers, Vadodara, March 8th 2014.

C4: Under preparation

(8) Ami Patel, Prof. S.K.Shah, “Real time implementation of power system stabilizer for SMIB system”, IEEE International conference on Electrical, Computer and Communication Technologies, Feb 22-24, 2017

(9) Ami Patel, Prof. S.K.Shah, “Multiobjective optimization of Single machine infinite bus system for power system stabilization”, IEEE International conference on communications, ICC 2017, May 21-25, 2017

APPENDIX D

PHOTOGALLERY

Following is the photographs of the developed graphical user interface.

The summary of work is presented in the form of MMAC having a graphical user interface shown in Fig. D.1 to D.8. The thesis & synopsis are also embedded in the software for on-line help. The graphical user interface shows with various option menus for robust multiple model adaptive controllers.

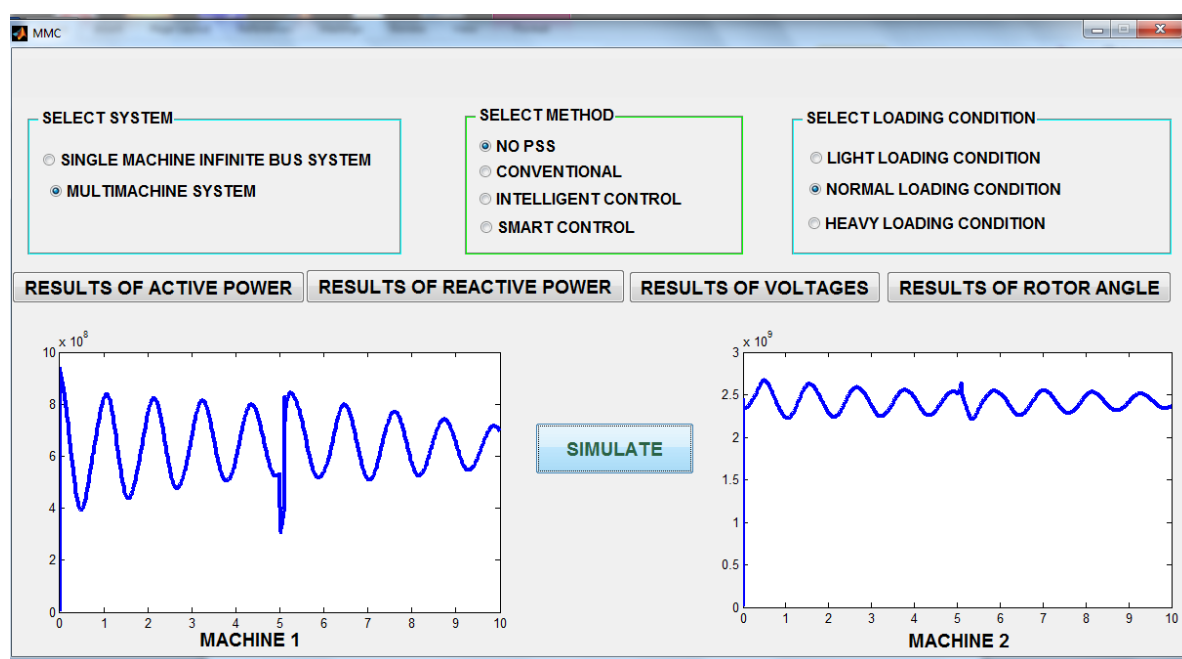


Fig. D. 1 GUI for Controller performance for Normal Loading Condition

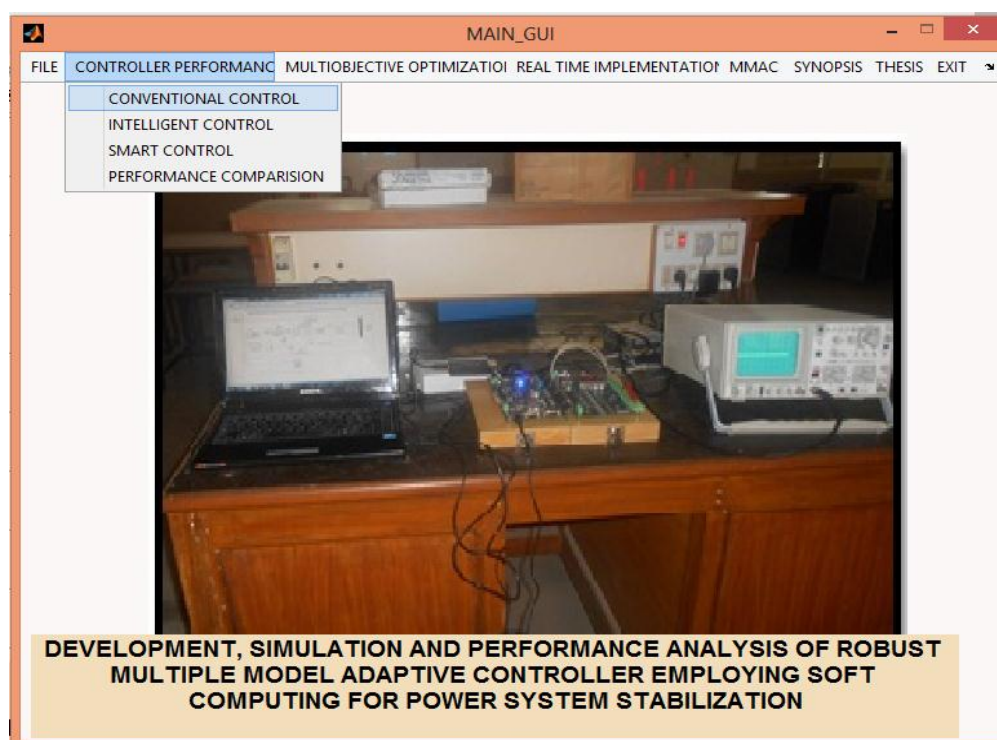


Fig. D. 2 Main Layout 1 of Graphical User Interface

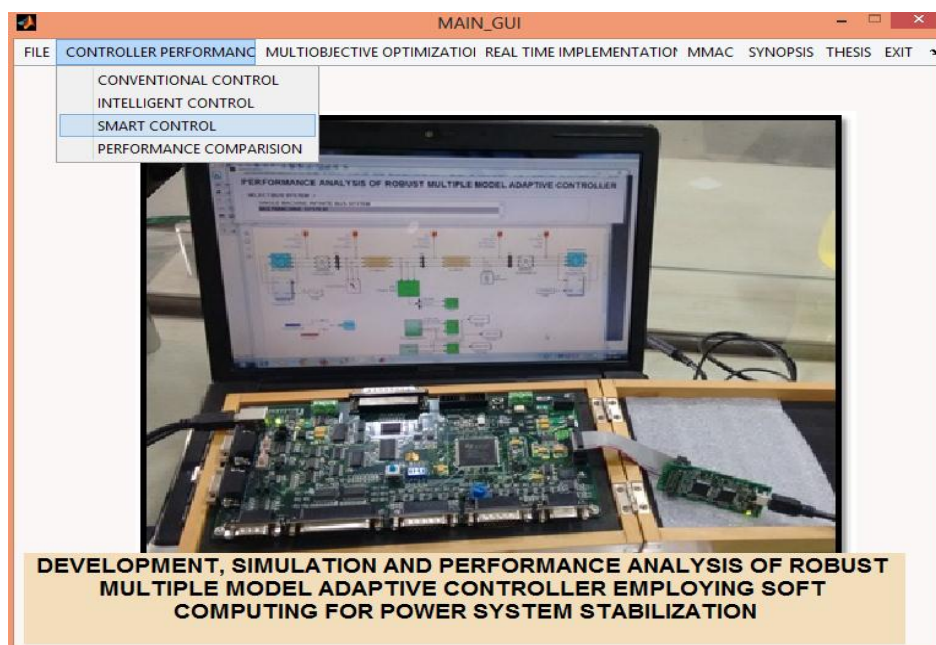


Fig. D. 3 Main Layout 2 of Graphical User Interface

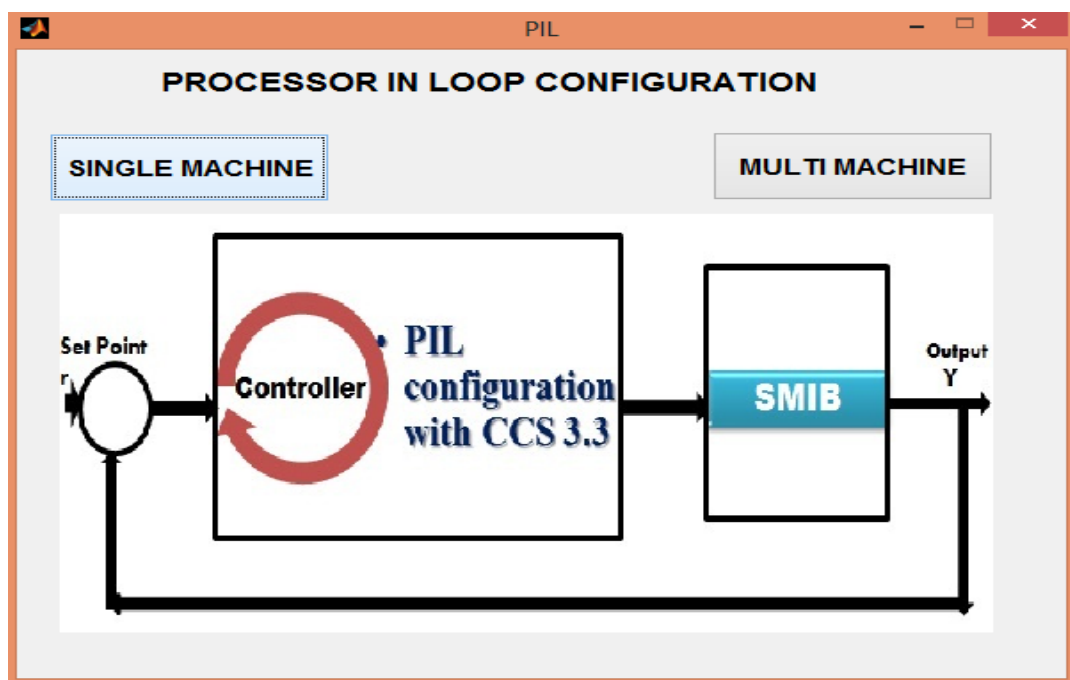


Fig. D. 4 GUI for Processor in loop configuration for SMIB system

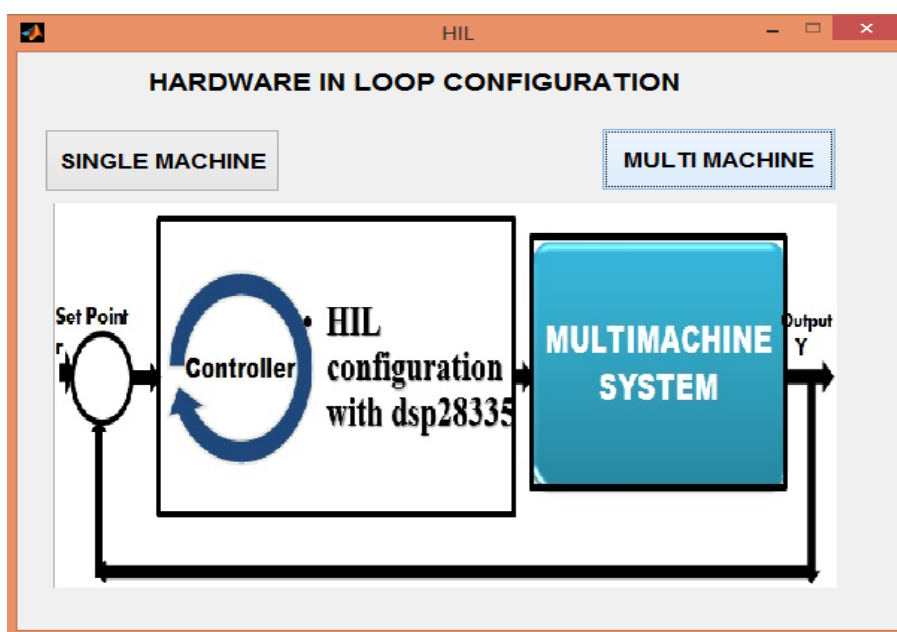


Fig. D. 5 Hardware in loop configuration for multimachine system

APPENDIX E

SOFT COMPUTATIONAL FIELDS

It provides a comprehensive study of the work done by the researchers using soft computational fields by using MATLAB Simulink for the design of single/Multi Machine system. The general preview of the soft computational fields such as: Fuzzy logic, Artificial Neural Network, Genetic Algorithm and Particle Swarm Optimization provided in this chapter with reference to observer, estimator and Controllers. It also describes the software tools available for development of FUZZY, ANN models, genetic algorithm and particle swarm optimization for parameter optimization and to carry out their simulation study. SIMULINK is used for testing the performance of the compensator.

E.1 Introduction

Soft computational fields are an artificial intelligence which relies on the algorithms such as fuzzy systems, neural networks genetic algorithm and particle swarm optimization. This chapter describes these methods and tools used for the development.

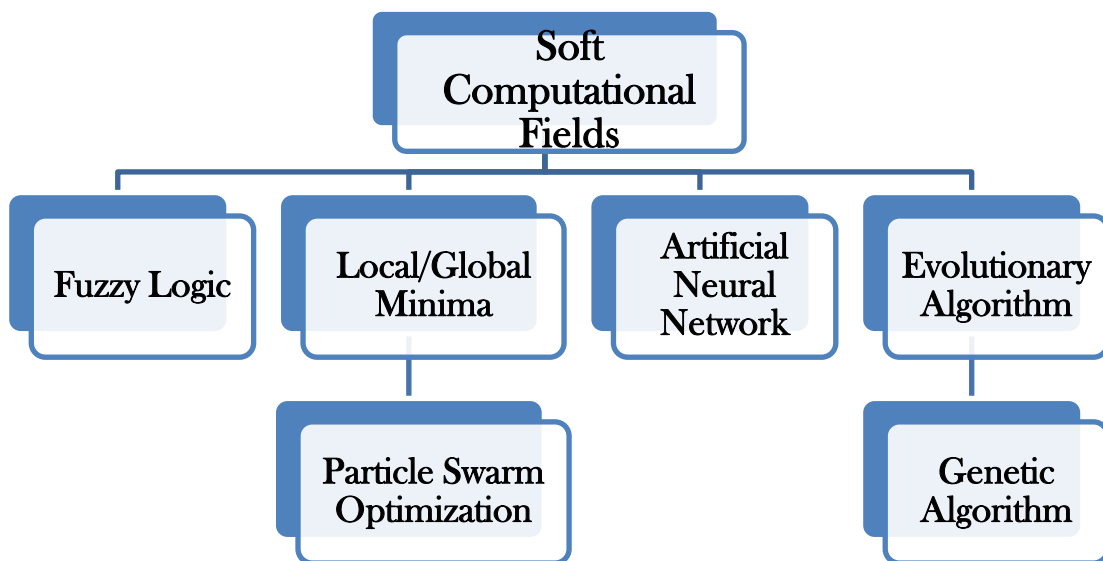


Fig. E. 1 Soft Computational Fields

E.2 MATLAB development tools

E.2.1 Main features and capabilities of MATLAB

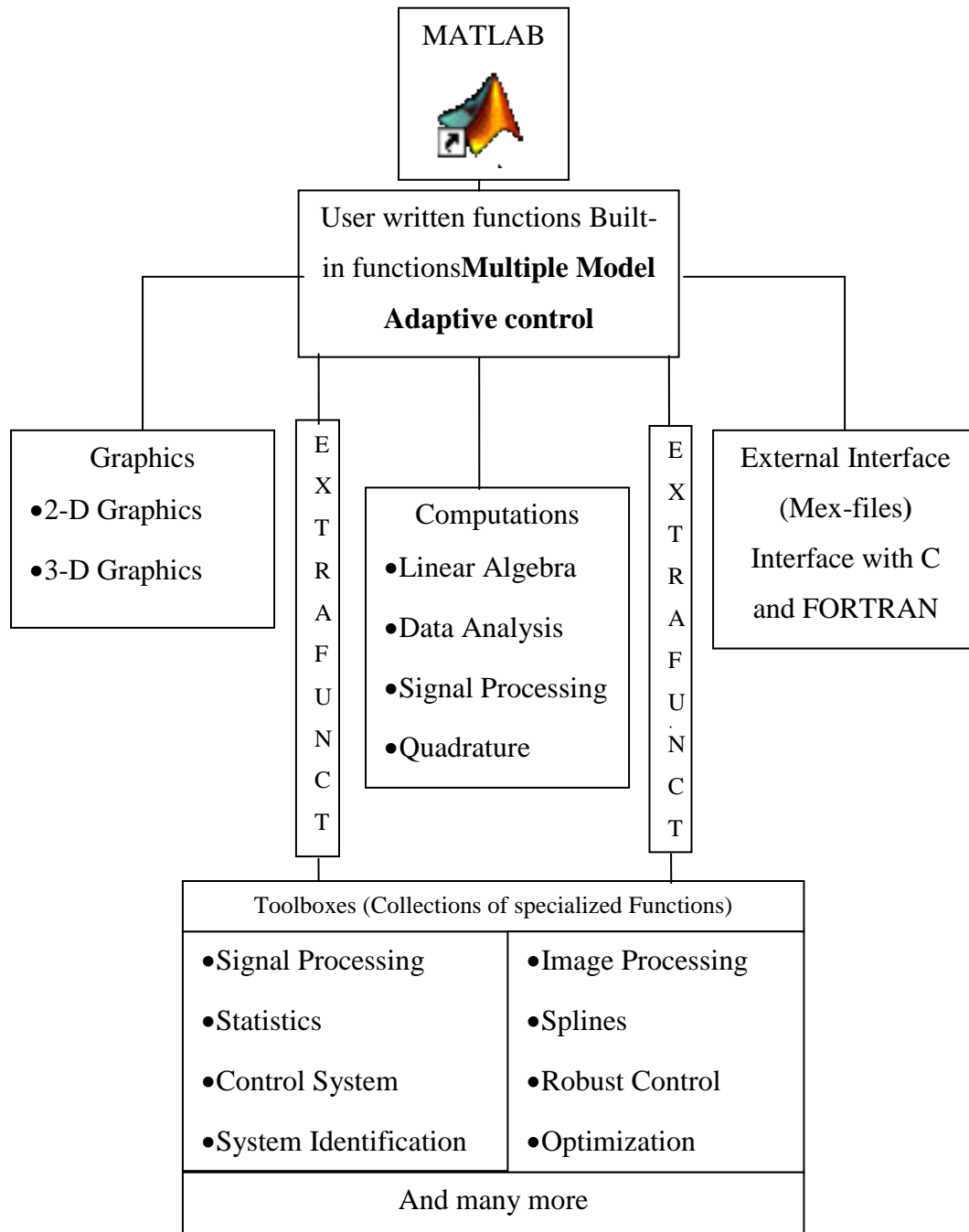


Fig. E. 2 Schematic diagrams of MATLAB's main features

When the MATLAB desktop appears, containing tools (graphical user interfaces) for managing files, variables, and applications associated with MATLAB. Default configuration of the MATLAB desktop as shown in Fig E.3.

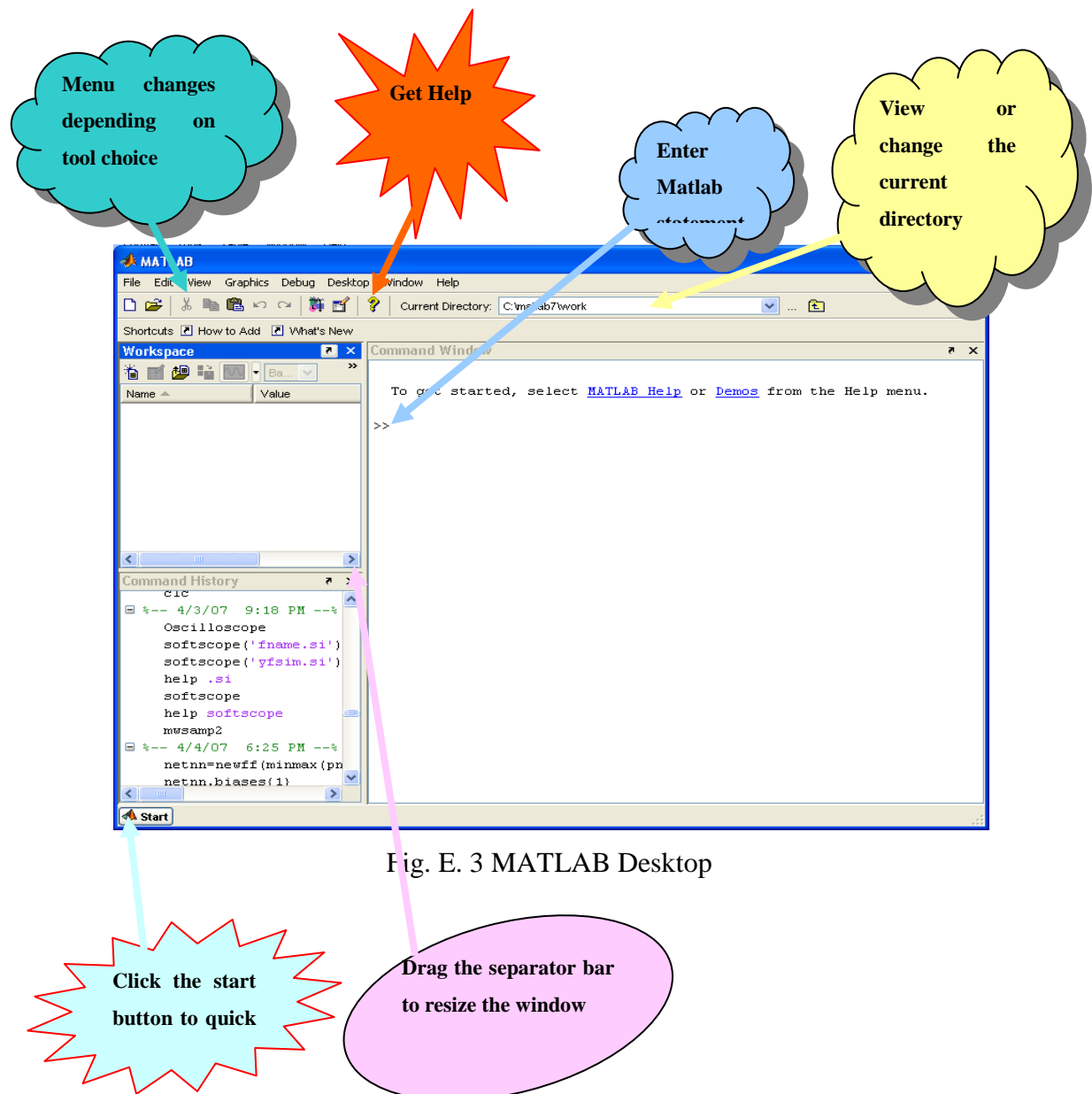


Fig. E. 3 MATLAB Desktop

Command window is the main window. It is characterized by the MATLAB command prompt “>>”. Command Window is used to enter variables and run functions and M-files as shown in Fig E.4.

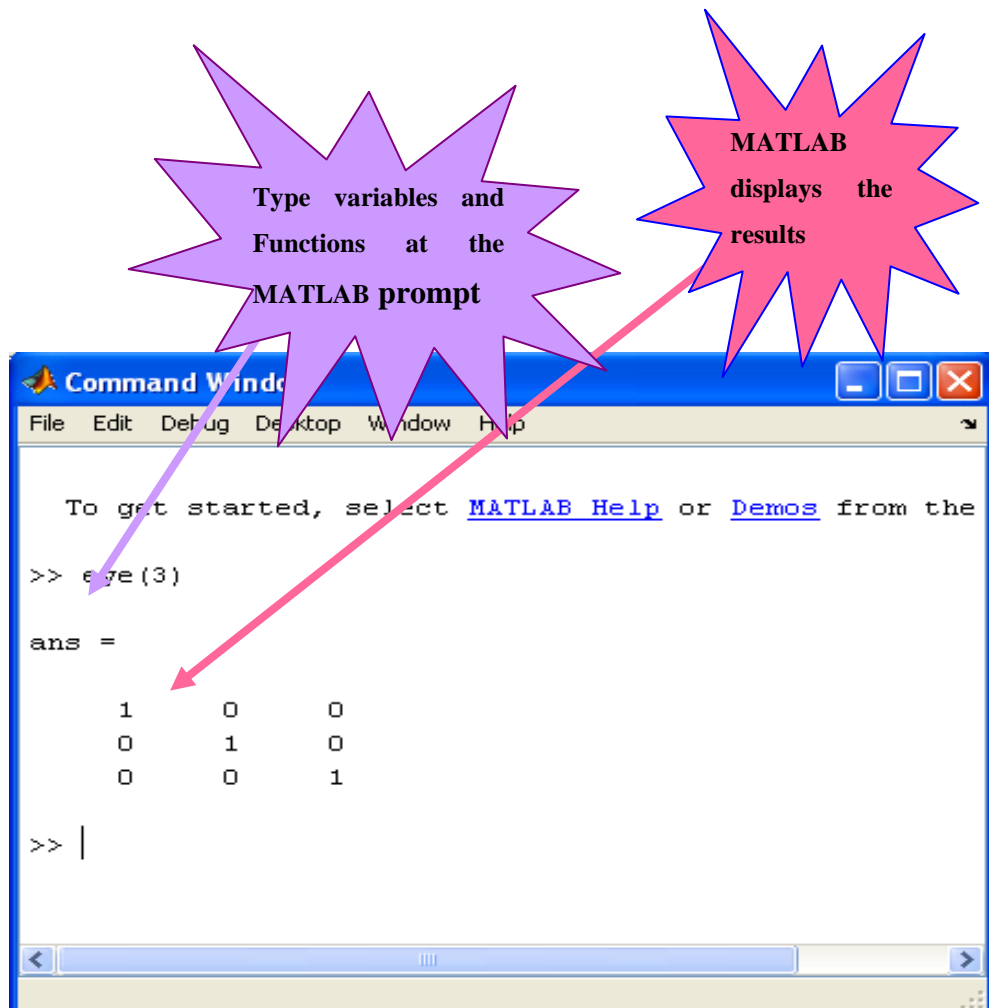


Fig. E. 4 Command Window

E.2.2 Simulink

Simulink is a software package for modeling, simulating, and analyzing dynamic systems. This package provides linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. Systems can also be multirate, i.e., have different parts that are Simulink has following features:

- Availability of extensive library of different blocks.

Models can be grouped to hierarchies to create a simplified view of components of subsystems. A Simulink block diagram model is a graphical representation of a mathematical model of a dynamic system which is shown by Fig E.5.

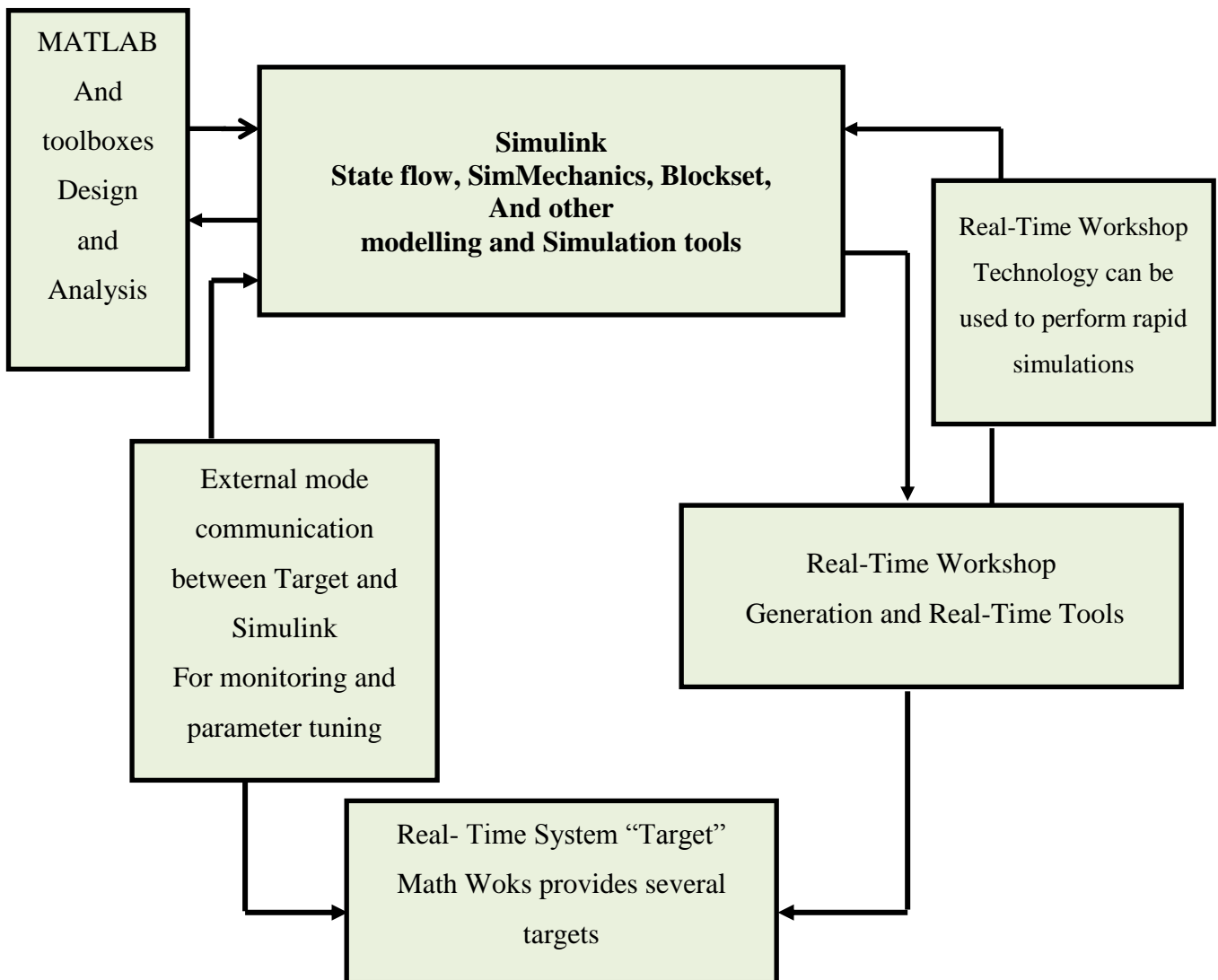


Fig. E. 5 Facilities of Simulink

Simulink provides a richest of modeling capabilities for dynamic systems, which can further be extended by domain specific products such as State flow for event driven systems, SimMechanics for modeling physical systems, and many Block sets such as the DSP Blockset

for signal processing. At any point during the design cycle, it can be use the power of MATLAB and the many toolboxes to analyze the simulation or real-time results or improve designs. Using Simulink:

- ✚ Build a block diagram.
- ✚ Simulate the system's behaviour

Evaluate its performance and refine the design

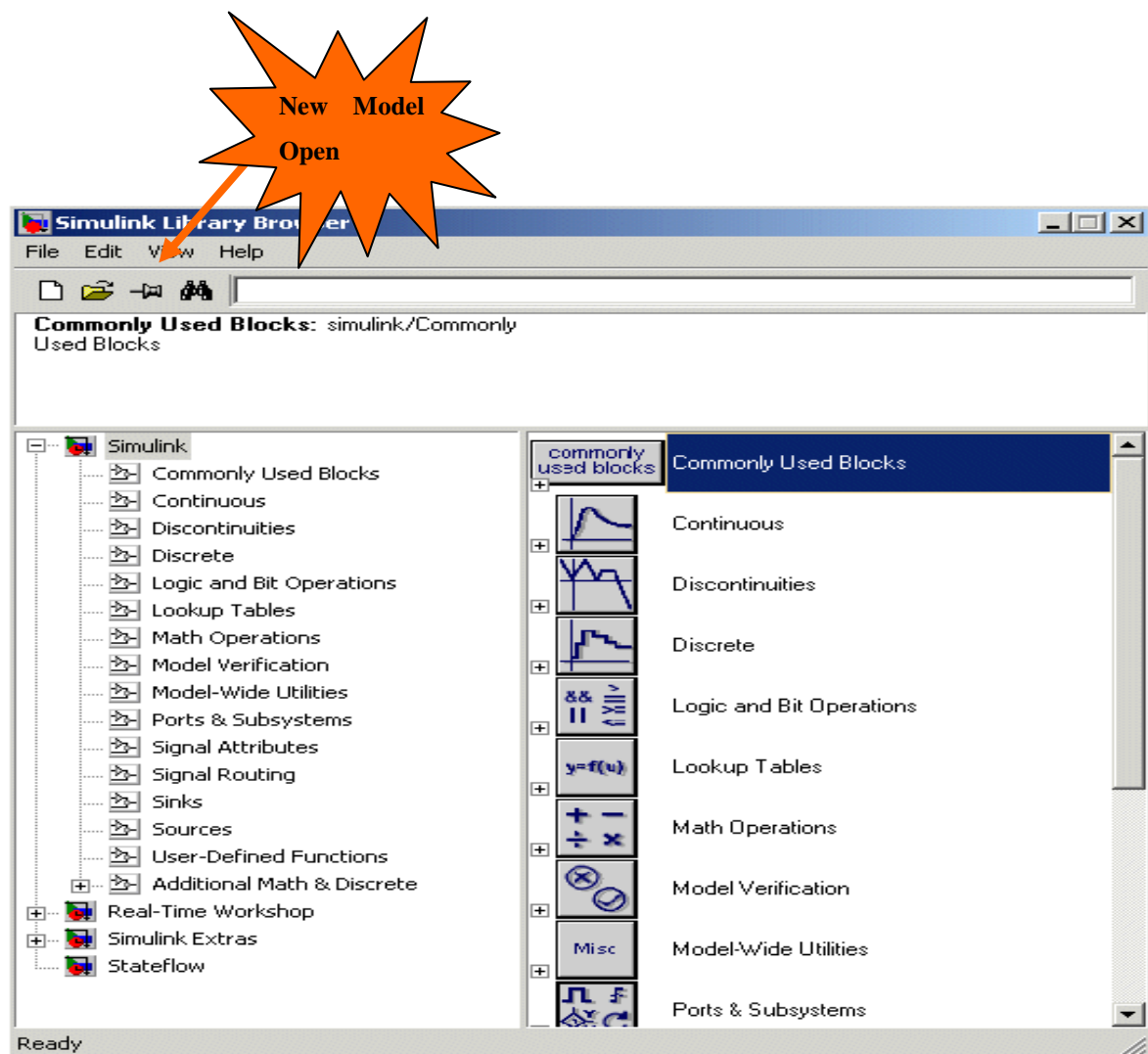


Fig. E. 6 Simulink library browser

E.2.3 Toolboxes

Fuzzy Logic toolbox

Table E. 1 Functions used to create fuzzy system

Functions	Description
Addmf	Add a membership function to an FIS
Addrule	Add a rule to an FIS
Addvar	Add a variable to an FIS
Evalfis	Perform fuzzy inference calculations
Newfis	Create new FIS
Trimf	Triangular membership function

Fuzzy Logic Toolbox™ provides MATLAB functions, graphical tools, and a Simulink block for analyzing, designing, and simulating systems based on fuzzy logic. The product guides you through the steps of designing fuzzy inference systems. Functions are provided for many common methods, including fuzzy clustering and adaptive neuro-fuzzy learning. Fuzzy inference blocks used in Simulink and simulate the fuzzy systems within a comprehensive model of the entire dynamic system [4].

Artificial Neural Network Toolbox

Table E. 2 Functions used from ANN toolbox

Functions	Description
newff	Create feed forward backpropagation network
Purelin	Linear transfer function
tansig	Hyperbolic tangent sigmoid transfer function
Traingd	Gradient descent backpropagation
sim	Simulation of Simulink model
gensim	Generate Simulink block simulate a neural network
Train	Trains a network according to NET.trainFcn and NET.trainParam

Neural Network Toolbox™ provides functions and apps for modeling complex nonlinear systems that are not easily modeled with a closed-form equation [5]. Neural Network Toolbox supports supervised learning with feed forward, radial basis, and dynamic networks. It also supports unsupervised learning with self-organizing maps and competitive layers. With the toolbox you can design, train, visualize, and simulate neural networks. Neural Network Toolbox can be used for applications such as data fitting, pattern recognition, clustering, time-series prediction, and dynamic system modeling and control. To speed up training and handle large data sets, you can distribute computations and data across multicore processors, GPUs, and computer clusters using Parallel Computing Toolbox™

Global Optimization Toolbox

Global Optimization Toolbox provides methods that search for global solutions to problems that contain multiple maxima or minima. It includes global search, multistart, pattern search, genetic algorithm, and simulated annealing solvers. You can use these solvers to solve optimization problems where the objective or constraint function is discontinuous, discontinuous, and stochastic, does not possess derivatives, or includes simulations or black-box functions with undefined values for some parameter settings.

Table E. 3 Functions used for global optimization toolbox

Functions	Description
createoptimProblem	Create optimization problem structure
CustomStartPointSet	User-supplied start points
GlobalOptimSolution	Optimization solution
GlobalSearch	Find global minimum
MultiStart	Find multiple local minima
RandomStartPointSet	Random start points

Genetic algorithm and pattern search solvers support algorithmic customization. You can create a custom genetic algorithm variant by modifying initial population and fitness scaling options or

by defining parent selection, crossover, and mutation functions. You can customize pattern search by defining polling, searching, and other functions [6].

✚ Optimization Problem Setup

Choose solver, define objective function and constraints, and compute in parallel

✚ Global or Multiple Starting Point Search

Multiple starting point solvers for gradient-based optimization, constrained or unconstrained

✚ Direct Search

Pattern search solver for derivative-free optimization, constrained or unconstrained

✚ Genetic Algorithm

Table E. 4Genetic algorithm Functions

Functions	Description
gab	Find the minimum of a function using the genetic algorithm
gaoptimget	Get values of a genetic options structure
gaoptimset	Create a genetic algorithm options structure
gatool	Open the genetic algorithm tool

Genetic algorithm solver for mixed-integer or continuous-variable optimization, constrained or unconstrained

✚ Particle Swarm :Particle swarm solver for derivative-free unconstrained optimization or optimization with bounds

✚ Simulated Annealing : Simulated annealing solver for derivative-free unconstrained optimization or optimization with bounds

✚ Multiobjective Optimization:Pareto sets via genetic algorithm with or without constraints (2).

E.3 Fuzzy logic control system

Fuzzy logic is derived from fuzzy set theory dealing with reasoning that is approximate rather than precisely deduced from classical predicate logic. It can be thought of as the application side

of fuzzy set theory dealing with well thought out real world expert values for a complex problem [7]. In this context, FL is a problem-solving control system methodology that lends itself to implementation in systems ranging from simple, small, embedded micro-controllers to large, networked, multi-channel PC or workstation-based data acquisition and control systems. The Fuzzy Logic tool was introduced in 1965, also by LOTFI ZADEH, and is a mathematical tool for dealing with uncertainty. It offers to a soft computing partnership the important concept of computing with words'. It provides a technique to deal with imprecision and information granularity. The fuzzy theory provides a mechanism for representing linguistic constructs such as "many," "low," "medium," "often," "few." In general, the fuzzy logic provides an inference structure that enables appropriate human reasoning capabilities [8].

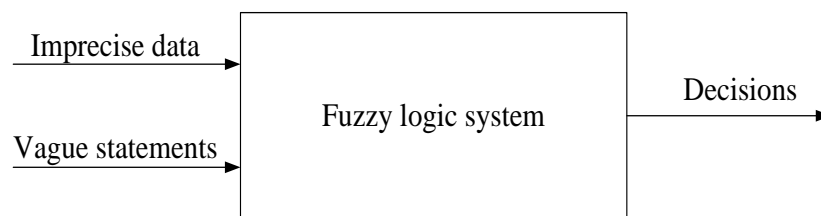


Fig. E. 7 Basic structure of fuzzy logic system

Fuzzy logic means approximate reasoning, information granulation, computing with words and so on. Ambiguity is always present in any realistic process. Fuzzy logic provides an inference structure that enables the human reasoning capabilities to be applied to artificial knowledge-based systems [9]. Fuzzy logic provides a means for converting linguistic strategy into control actions and thus offers high-level computation.

Fuzzy logic provides mathematical strength to the emulation of certain perceptual and linguistic attributes associated with human cognition, whereas the science of neural networks provides a new computing tool with learning and adaptation capabilities. The theory of fuzzy logic provides an inference mechanism under cognitive uncertainty; computational neural networks offer exciting advantages such as learning, adaptation, fault tolerance, parallelism, and generalization [10]. A fuzzy logic system which accepts imprecise data and vague statements such as low, medium, high and provides decisions which is shown in Fig E.7

E.3.1 Design of fuzzy logic controller

A typical fuzzy system consists of a rule base, membership functions and an inference system. The design of a fuzzy logic controller which consist of a fuzzification interface, a knowledge base, decision making logic, and a defuzzification interface.

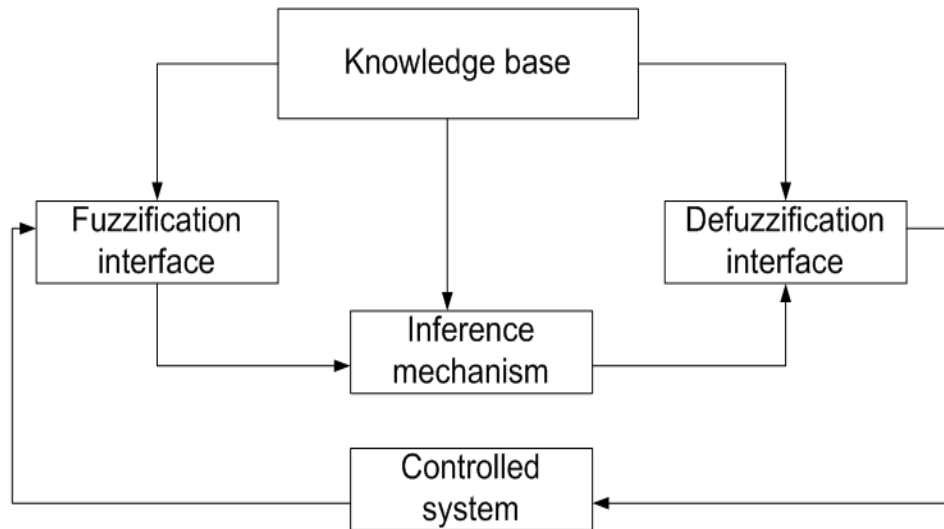


Fig. E. 8 Components of Fuzzy controller

In fuzzification the value of input variables are measured and then converts input data into suitable linguistic values which may be viewed as label fuzzy sets. The knowledge base comprises knowledge of application domain and attendant control goals. It consists of a database and linguistic control rule base. The database provides the necessary definitions, which are used to define linguistic control rules and fuzzy data manipulation in an FLC. The rule base are the control strategy of the FIS system. The FIS system formulates suitable rules and based upon the rules the decision is made. There are two methods MAMDANI and SUGENO for fuzzification [11]. After fuzzification, the fuzzy output is converted into crisp output by defuzzification method. The different methods of defuzzification are weighted average method, mean of maxima method and centroid method, centre of sums method etc.

(2) Paper entitled, “Development of Real time controller of a Single Machine Infinite Bus system with PSS”, *International Journal of Electrical Engineering (IJEE)*, ISSN 2321-600X, Volume 2, Issue 9, September 2014, Impact Factor: 1.318.

E.3.2 Fuzzy implication methods

Mamdani or Max-min method

The max-min algorithm operates on each rule (min fashion) and combining all the rule (max fashion). In the min composition, for each rule, the algorithm matches the membership degrees to the antecedent membership function, and finds the minimum. The minimum function is the equivalent of the AND logical function. For each output variable, a matrix is constructed where each row corresponds to a rule and each column corresponds to a crisp value in the fuzzy set. In the max composition, the combined output fuzzy subset is constructed by taking the maximum over all of the fuzzy subsets assigned to the output variable by the inference rule. The maximum of each column is then calculated, which yields a composite or inferred membership function that is then passed to the defuzzification procedure.

Sugeno Method

The Takagi-sugeno fuzzy model (Takagi and sugeno 1985) uses crisp functions as the consequences of the rules. This is the difference between the mamdani and sugeno method. The antecedent of each rule is a set of fuzzy propositions connected with the AND operator (for more than one input). The consequent of each rule is a crisp function of the input vector. By means of the fuzzy sets of the antecedent propositions the input domain is softly partitioned into smaller regions where the mapping is locally approximated by the crisp functions. Takagi- sugeno rule aggregation and their effects considerably differ from the Mamdani method. One variation of the Takagi- sugeno inference system uses the weighted mean criterion to combine all the local representations in a global approximate.

E.4 Artificial Neural Network

Almost all information-processing needs of today are met by digital computers. Neural networks are constructed with neurons that connected to each other [12]. A more formal definition of an ANN according to Haykin is: “A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use” [12].

E.4.1 The Biological Inspiration

The brain is principally composed of a very large number of neurons, massively interconnected. Each neuron is a specialized cell which can propagate an electrochemical signal. The neuron has a branching input structure (the dendrites), a cell body, and a branching output structure (the axon). The axons of one cell connect to the dendrites of another via a synapse. Neuron is activated, it fireman electrochemical signal along the axon. This signal crosses the synapses to other neurons, which may in turn fire [13].

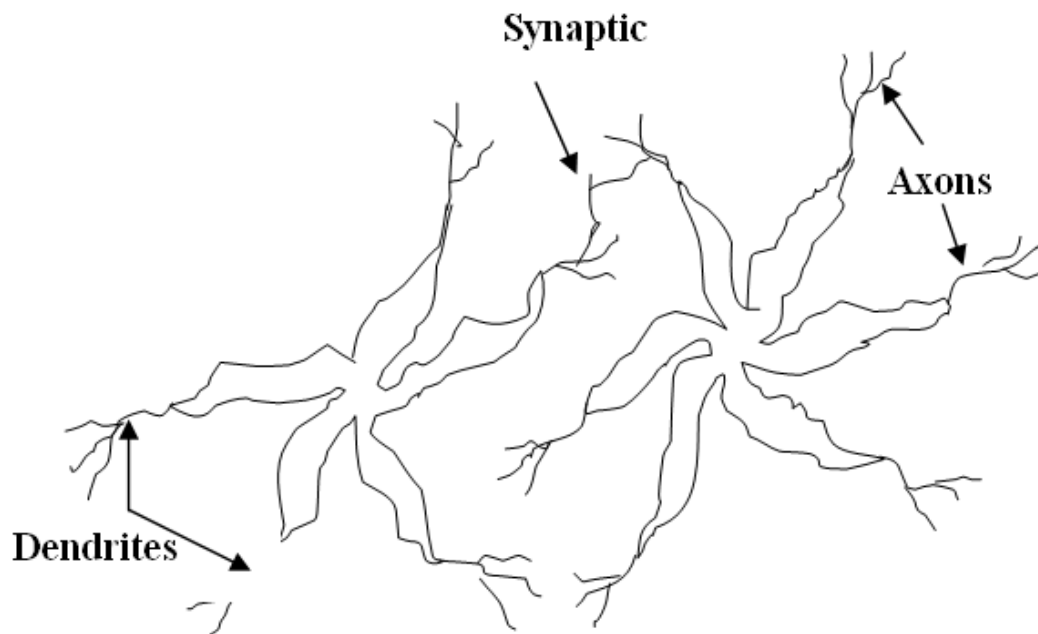


Fig. E. 9The Biological Neurons

It receives a number of inputs either from original data, or from the output of other neurons in the neural network. Each input comes via a connection that has strength (weight); these weights correspond to synaptic efficacy in a biological neuron. Each neuron also has a single threshold value. The weighted sum of the inputs is formed, and the threshold subtracted, to compose the activation of the neuron (also known as the post-synaptic potential, or PSP, of the neuron). The activation signal is passed through an activation function (also known as a transfer function) to produce the output of the neuron [14] [15].

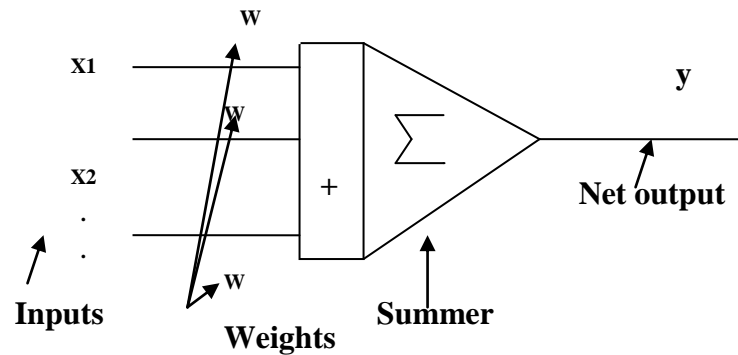


Fig. E. 10 Simple model of an Artificial Neuron

E.4.2 Single layer and multi-layer networks

Single layer networks

For single layer neural network, the output signals of the neurons in the first layer are the output signals of the network.

Here each neuron adjusts its weights according to what output was expected of it, and the output it gave.

Multi-layer networks

Multilayerperception's are feed forward nets with one or more layers of nodes between the input and output nodes. Multilayer feed forward networks normally consist of three or four layers; there is always one input layer and one output layer and usually one or more hidden layers. The term input layer neurons are a misnomer; no sigmoid unit is applied to the value of each of these neurons. Their raw values are fed into the layer downstream the input layer (the hidden layer). Once the neurons for the hidden layer are computed, their activations are then fed downstream to the next layer, until all the activations eventually reach the output layer, in which each output layer neuron is associated with a specific classification category [16].

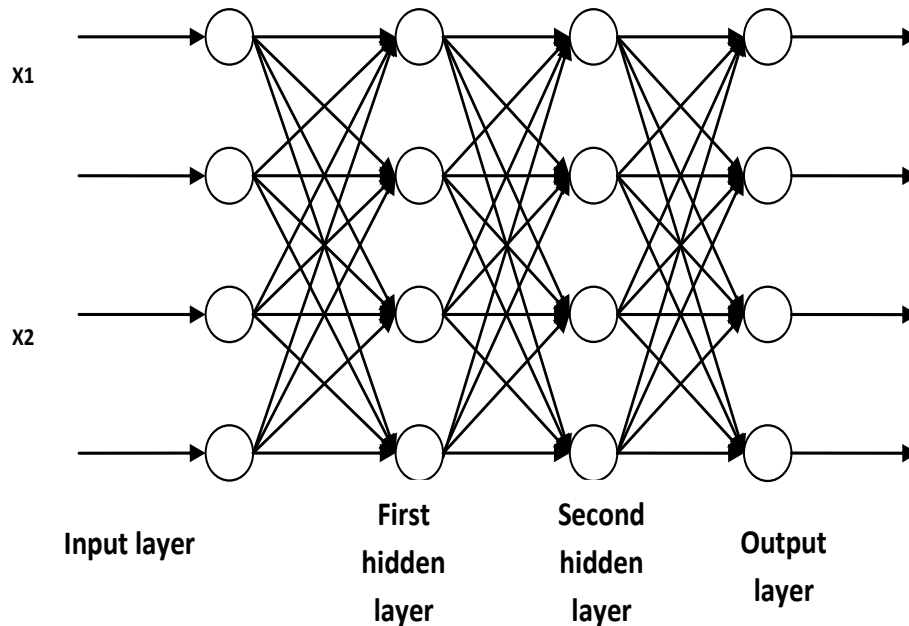



Fig. E. 11 Multilayer Networks

E.4.3 Types of neural network learning

“The algorithmic process of weight adjustments is called learning rule”.

They can be grouped into two groups:

1. Supervised learning and
2. Unsupervised learning.

 Supervised learning:

Supervised learning is a process of training a neural network by giving it examples of the task we want it to learn. i.e. it is a learning with a teacher. The way this is done is by providing a set of pairs of vectors (patterns), where the first pattern of each pair is an example of an input pattern that the network might have to process and the second pattern is the output pattern that the network should produce for that input which is known as a target output pattern for whatever input pattern.

Unsupervised learning:

It is the learning process in which changes in a network's weights and biases are not due to the intervention of any external teacher. Commonly changes are a function of the current network input vectors, output vectors, and previous weights and biases. In this the network is able to discover statistical regularities in its input space and automatically develops different modes of behavior to represent different classes of inputs (in practical applications some labeling is required after training, since it is not known at the outset which mode of behavior will be associated with a given input class). In this type of learning due to absence of desired output it is difficult to predict what type of features network will extract [17].

E.4.4 Procedure for ANN Implementation

The general steps can be summarized as follows:

1. Analyze the problem and find whether it has sufficient elements for a neural network.
2. If the ANN is to represent a static function then a three layer feed forward network should be sufficient. For a dynamic function, select either a recurrent neural network or a time delayed network.
3. Select an input and output signals. For a feed forward network, select the hidden layer neurons.
4. Select generally a sigmoid transfer network for unipolar output and a hyperbolic tan function for bipolar output.
5. Select a development system such as Neural Network toolbox in MATLAB.
6. Select appropriate learning coefficient (η) and momentum factor (μ).
7. Select an acceptable training error ξ and a no. Of epochs.
8. After the training is complete with all patterns, test the network performance.

E.5 Genetic Algorithm

Genetic algorithms (GAs), which uses the concept of Darwin's theory, have been widely introduced to deal with nonlinear control difficulties and to solve complicated optimization

problems. Darwin's theory basically stressed the fact that the existence of all living things is based on the rule of 'survival of the fittest'. In the theory of evolution, different possible solutions to a problem are selected first to a population of binary strings encoding the parameter space. The selected solutions undergo a parallel global search process of reproduction, crossover and mutation to create a new generation with the highest fitness function [11]. Genetic algorithms (GAs) are computer-based search techniques patterned after the genetic mechanisms of biological organisms which have allowed such organisms (GAs), to adapt and flourish in changing, highly competitive environments for millions of years.

E.5.1 Solution Representation: The Chromosome Structure

Typically a bit-string or a string of some other kind is used to represent a solution in evolutionary computing. The string data structure is most similar to the natural chromosome and therefore the string can be manipulated in ways similar to natural chromosomes.

Binary Encoding: -

The most commonly used representation of chromosomes in the GA is that of the single-level binary string. Here, each decision variable in the parameter set is encoded as a binary string and these are concatenated to form a chromosome. It is easy to implement and understand. It supports implicit parallelism. In binary encoding, every chromosome is a string of bits 0 or 1.

Real valued encoding: -

In real encoding, every chromosome is a sequence of some values. Values can be a real numbers, chars or any objects.

E.g. - Chromosome A: 2.5 8.9 0.68 7.2

Chromosome B: SWEFGRDJHFHF

Efficiency of the GA increases, as there is no need to convert chromosomes to phenotypes before each function evaluation.

Less memory is required as efficient floating-point internal computer representations can be used directly;

There is no loss in precision by discretization to binary or other values.

There is greater freedom to use different genetic operators. The use of real-valued encoding is described in detail in [18]

✚ Permutation encoding: -In this type of encoding every chromosome is a string of numbers i.e. has real sequence in it, which represents position in a sequence and can be used in the ordering problems, such as travelling salesman problem (distance optimization) or task ordering problem (efficiency optimization).

E.g. - Chromosome A: 5 4 9 1 2 7 0 3 4 5 8

✚ Tree encoding: -

In the tree encoding every chromosome is a tree of some objects, such as functions or commands in programming language. It is mainly used for evolving programs.

E.5.2 General Rules to Set Parameters of Genetic Algorithm

1. Population size: - It influences amount of search points in every generation. There is always a trade-off between diversity of population and computation time. The more population size in the Gas will increase the efficiency of searching, but it will time consuming. When the population is less GA may converge in too few generations to ensure a good solution. The population size usually ranges from five to ten times the number of searched variables. [20]
2. Crossover Probability: - The crossover probability controls the rate at which solutions are subjected to crossover i.e. influences the efficiency of exchanging information. The higher the crossover probability the quicker the new solutions are generated & lower crossover rate may stagnate the search due to loss of exploration power. Typical values of the crossover probability are in the range 0.5-1.0.
3. Mutation Probability: - A large value of the mutation probability transforms the GA into a purely random search algorithm by eliminating the results of reproduction & crossover. While a certain value of the mutation probability is required to prevent the convergence of the GA to sub optimal solutions. A typical value of the mutation probability is in the range 0.0-0.1.
4. Chromosome length: - It influences the resolution of the searching result. The GAs with longer chromosome length will have the higher resolution, but it will increase the search space.

5. Number of Generations: - which influences the searching time and searching result. The GAs with larger search space and less population size, it needs more generations for a global optimum. Additionally, the set of the number of generations is dependent on the problem [21].

E.6 Particle Swarm Optimization

E.6.1 Basic of Particle Swarm Optimization

Particle swarm is a population-based algorithm. In this respect it is similar to the genetic algorithm. A collection of individuals called particles move in steps throughout a region. At each step, the algorithm evaluates the objective function at each particle. After this evaluation, the algorithm decides on the new velocity of each particle. The particles move, then the algorithm re-evaluates. The inspiration for the algorithm is flocks of birds or insects swarming. Each particle is attracted to some degree to the best location it has found so far, and also to the best location any member of the swarm has found. After some steps, the population can coalesce around one location, or can coalesce around a few locations, or can continue to move. The particle swarm algorithm begins by creating the initial particles, and assigning them initial velocities. It evaluates the objective function at each particle location, and determines the best (lowest) function value and the best location. It chooses new velocities, based on the current velocity, the particles' individual best locations, and the best locations of their neighbours. It then iteratively updates the particle locations (the new location is the old one plus the velocity, modified to keep particles within bounds), velocities, and neighbours. Iterations proceed until the algorithm reaches a stopping criterion [22].

E.6.2 General Procedure of PSO algorithm

The algorithm updates the swarm as follows. For particle i , which is at position $x(i)$:

1. Choose random subset S of N particles other than i .
2. Find $f_{best}(S)$, the best objective function among the neighbors, and $g(S)$, the position of the neighbor with the best objective function.
3. For u_1 and u_2 uniformly (0,1) distributed random vectors of length $nvars$, update the velocity
 - a.
$$v = W*v + y_1*u_1.*(p-x) + y_2*u_2.*(g-x).$$

- b. This update uses a weighted sum of:
 - c. The previous velocity v
 - d. The difference between the current position and the best position the particle has seen $p-x$
 - e. The difference between the current position and the best position in the current neighbourhood, $g-x$
- 4. Update the position $x = x + v$.
- 5. Enforce the bounds. If any component of x is outside a bound, set it equal to that bound.
- 6. Evaluate the objective function $f = \text{fun}(x)$.
- 7. If $f < \text{fun}(p)$, then set $p = x$. This step ensures p has the best position the particle has seen.
- 8. If $f < b$, then set $b = f$ and $d = x$. This step ensures b has the best objective function in the swarm, and d has the best location.
- 9. If, in the previous step, the best function value was lowered, then set $\text{flag} = \text{true}$. Otherwise, $\text{flag} = \text{false}$. The value of flag is used in the next step.
- 10. Update the neighborhood. If $\text{flag} = \text{true}$:
 - a. Set $c = \max(0, c-1)$.
 - b. Set N to $\text{minNeighbourhoodSize}$.
 - c. If $c < 2$, then set $W = 2*W$.
 - d. If $c > 5$, then set $W = W/2$.
 - e. Ensure that W is in the bounds of the InertiaRange option.
 - f. If $\text{flag} = \text{false}$:
 - g. Set $c = c+1$.
 - h. Set $N = \min(N + \text{minNeighbourhoodSize}, \text{SwarmSize})$.

E.7 Code Composer Studio

E.7.1 Integrated development environment (IDE)

Code Composer Studio is an integrated development environment (IDE) that supports TI's Microcontroller and Embedded Processors portfolio. Code Composer Studio comprises a suite of tools used to develop and debug embedded applications. It includes an optimizing C/C++

compiler, source code editor, project build environment, debugger, profiler, and many other features. The intuitive IDE provides a single user interface taking you through each step of the application development flow. Familiar tools and interfaces allow users to get started faster than ever before. Code Composer Studio combines the advantages of the Eclipse software framework with advanced embedded debug capabilities from TI resulting in a compelling feature-rich development environment for embedded developers.

Code Composer Studio is comprised of a suite of tools used to develop and debug embedded applications. It includes a compiler source code editor, project build environment, debugger, profiler and many other features. The intuitive IDE provides a single user interface taking you through each step of the application development flow. Familiar tools and interfaces allow users to get started faster than ever before and add functionality to their application [24].

E.7.2 DSP/BIOS

DSP/BIOS are a scalable, real-time kernel that is designed for applications that require real-time scheduling and synchronization, host-to-target communication, or real-time instrumentation. The DSP/BIOS kernel is packaged as a set of modules that can be linked into an application. It is integrated with Code Composer Studio Integrated Development Environment (IDE), requires no runtime license fees, and is fully supported by Texas Instruments. The kernel is also a key component of TI's expressDSP™ technology.

DSP/BIOS kernel enables you to develop and deploy sophisticated applications and eliminates the need to develop and maintain custom operating systems or control loops. Because multi-threading enables real-time applications to be cleanly partitioned, applications using DSP/BIOS kernel are easier to maintain and new functions can be added without disrupting real-time response. DSP/BIOS kernel provides standardized APIs across TMS320C2000™, TMS320C5000™ and TMS320C6000™ DSP platforms to support rapid application migration. Additionally, it includes configuration support for EVMs, DSKs, simulators, and some third-party boards. Existing configuration templates are easily adaptable to provide support for custom

boards and other third-party boards. DSP/BIOS kernel is integrated into the Code Composer Studio IDE. Code Composer Studio's kernel object viewer and real-time analysis provide powerful set of integrated tools specifically focused on debugging and tuning multitasking applications [25].

E.7.3 Analysis & Evaluation tools

The embedded link places a constraint that the algorithm must be developed in the Simulink environment. Thus the algorithm was developed using the embedded MATLAB function in Simulink. Using the embedded function feature, a custom block in Simulink is created which contains the algorithm. The algorithm was built onto the DSP using the CCS link available on MATLAB as explained in the previous section [26].

E.8 Concluding Remarks

The Theoretical background of soft computational fields such as Fuzzy Logic, Artificial Neural Network, Genetic Algorithm, Particle swarm optimization and Code composer studio is summarized and described. Toolboxes available for deploying soft computational fields in MATLAB, CCS v3.3 and used in our research work for the design and testing of proposed techniques are described in detail. Procedural steps to be followed in each trait are discussed in detail.