A Novel Methodology for Automated Synthesis of Mechatronic Systems Designs Using Bond-Graphs and Genetic Programming

Saheeb Ahmed Kayani, Afzaal M. Malik
Department of Mechanical Engineering, College of Electrical and Mechanical Engineering,
National University of Sciences and Technology, Islamabad, Pakistan
mafzmlk@ceme.edu.pk

Abstract

Automated synthesis refers to design of physical systems using any of the models proposed for machine intelligence like evolutionary computation, neural networks and fuzzy logic. Mechatronic systems are mixed or hybrid systems as they combine elements from different energy domains. These dynamic systems are inherently complex and capturing underlying energy behavior among interacting sub-systems is difficult owing to the variety in the composition of the mechatronic systems and also due to the limitation imposed by conventional modeling techniques unable to handle more than one energy domain. Bond-Graph modeling and simulation is an advanced domain independent, object oriented and polymorphic graphical description of physical systems. The universal modeling paradigm offered by Bond-Graphs is well suited for mechatronic systems as it can represent their multi energy domain character using a unified notation scheme. Genetic programming is one of the most promising evolutionary computation techniques. The genetic programming paradigm is modeled on Darwinian concepts of evolution and natural selection. Genetic programming starts from a high level statement of a problem's requirements along with a fitness criterion and attempts to produce a computer program that provides a solution to the problem. Combining unified modeling and analysis tools offered by Bond-Graphs with topologically open ended synthesis and search capability of genetic programming, a novel automated design methodology has been developed for generating mechatronic systems designs using an integrated synthesis, analysis and feedback scheme which comes close to the definition of a true automated invention machine. This research paper is a brief introduction to all concepts associated with automated design of mechatronic systems using Bond-Graphs and genetic programming and explains the novel automated design methodology with a design example.

Keywords: Automated Synthesis, Mechatronic Systems, Bond-Graphs, Genetic Programming

Introduction

In this section we briefly introduce certain concepts prerequisite for understanding the functionality and appreciating the novelty of the automated mechatronic systems design methodology presented in following section.

Mechatronic systems

Typically mechanical systems with electronic control schemes have been categorized as mechatronic systems [1]. A servo system comprising of a voltage source, DC motor and a mechanical load driven through a transmission is an example of a mechatronic system based on this classification [2]. This long standing concept can now be extended. A generic mechatronic system is a multi domain dynamic collection of three basic interacting sub-systems: sensors, microprocessors or computers and actuators. On a functional design or topological level it is a hybrid combination of elements from different energy domains as varied as fluid power, hydraulic, pneumatic, electronic, acoustic, thermal and magnetic systems etc. put together to develop any of the three basic sub-systems.

Essentially whenever we try to model a mechatronic system we experience sheer variety attributable to the composition and very nature of the said system. There can be many representations of mechatronic systems but with the help of aforementioned statement we can realize a generic description for any mechatronic system like the one appearing in Figure 1. Here we are interested in over all system model (physical and mathematical) and dynamic response of interacting elements residing in different energy domains [3].

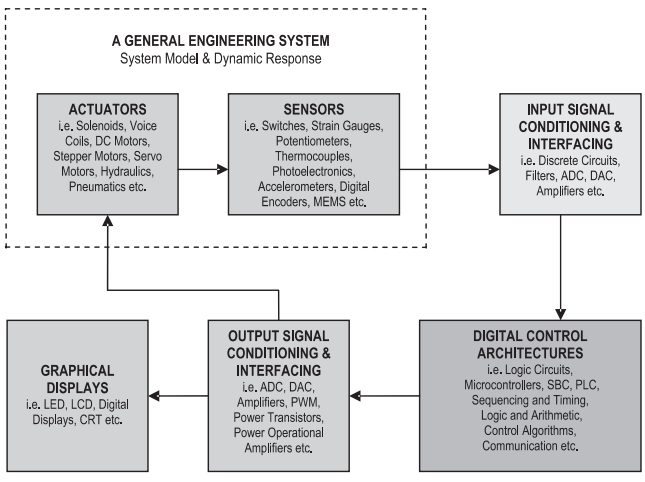

Fig. 1. Description of a generic mechatronic system

Corresponding Author: Afzaal M. Malik
NUST Publishing, © (2008), ISSN: 2070-9900

To design and analyze a mechatronic system and its multi energy sub-systems we need an inter domain tool. Analogies among interacting physical systems and energy exchange (storage, transport and dissipation) taking place at the proximity of each element and each sub-system provides a basis for Bond-Graphs based modeling [4].

Bond-Graph modeling

Bond-Graphs were invented by Henry M. Paynter (1923-2002) in 1950s while working as a civil engineer in hydroelectric power plants and later as a faculty member at MIT [5], [6]. They were further developed by Dean C. Karnopp, Ronald C. Rosenberg, Jan J. van Dixhoorn, Jean U. Thoma and Peter C. Breedveld during the period 1960-1990 and fundamental concepts of the technique were established, tested and extended [7], [8], [9]. Fifty years on the technique remains a high end modeling approach and a tool of choice for multi energy domain systems designers with great potential for newer application avenues.

Bond-Graphs offer an advanced domain independent, object oriented and polymorphic description of interacting sub-systems which are combined in an over all system model that can be simulated using computers [10]. Bond-Graph modeling is a port based technique where ports are places at which power can flow between sub-systems [11]. Physical systems with one or more ports are called multi ports. Bond-Graphs are labeled di-graphs where the edges are called bonds and represent the bilateral signal flow of the power conjugate variables effort *e(t)* and flow *f(t)* [2]. Each bond carries power represented by the product of effort and flow variables. Time integral of effort momentum *p(t)* and time integral of flow displacement *q(t)* are the only two other variables (referred to as energy variables) required for system modeling with Bond-Graphs. An example of these variables from mechanical and electrical domains seems appropriate here. For mechanical translational systems power variables are effort *e(t)*:force *F(t)* and flow *f(t)*:velocity *V(t)* where as energy

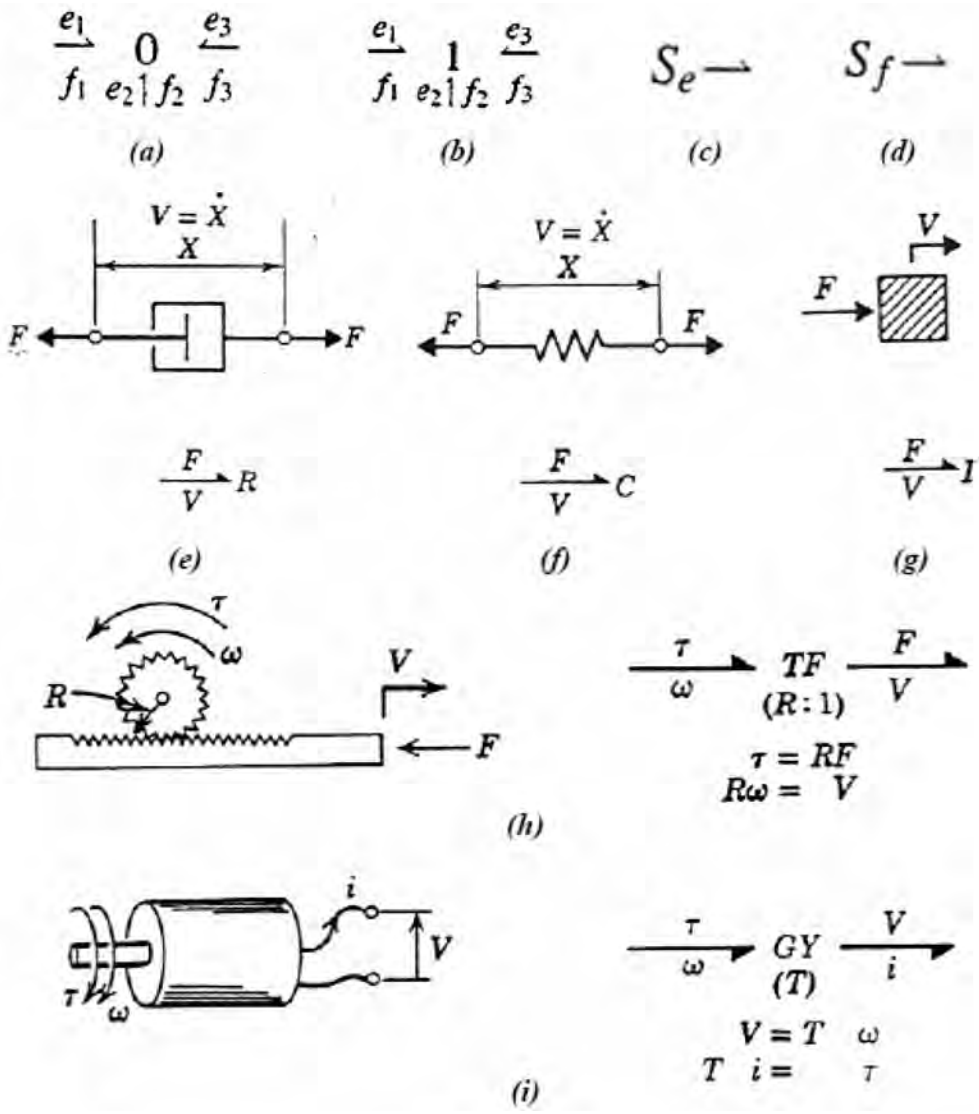

Fig. 2. Fundamental Bond-Graph Elements (a) *0* Junction (b) *1* Junction (c) Source of Effort *Se* (d) Source of Flow *Sf* (e) Resistor *R* for Mechanical Translation Systems (f) Capacitor *C* for Mechanical Translation Systems (g) Inertia *I* for Mechanical Translation Systems (h) Transformer *TF* (i) Gyrator *GY*

variables are momentum *p(t)*:momentum *P* and displacement *q(t)*:displacement *X* respectively. For electrical systems power variables are effort *e(t)*:voltage *E(t)* and flow *f(t)*:current *I(t)* where as energy variables are momentum *p(t)*:flux linkage variable λ and displacement *q(t)*:charge *Q* respectively [12].

The basic elements used in Bond-Graph modeling are five 1-port elements namely resistor *R*, capacitor *C* and inertia *I* (which dissipate and store energy respectively), source of effort *Se* and source of flow *Sf* (which represent sources of energy in physical systems), two 2-port elements which can scale energy up or down (transformer *TF*) and transform energy from one domain to another (gyrator *GY*) and two 3-port *0* and *1* junctions depicting physical or topological constraints like inter domain connections [13].

Causality establishes the computational direction of the effort variable. It is represented by a short perpendicular line made at one end of a bond. Sources of effort and flow have fixed causality (*Se*: effort out causality, *Sf*: flow out causality), energy storing elements *C* and *I* have preferred integral causality while resistor *R* has arbitrary or free causality. The junction elements have constrained causality (only one effort out causality at a *1* junction and one flow out causality at a *0* junction). Same is the case with transformer *TF* (only one effort out causality and one flow out causality) and gyrator *GY* (either two effort out causalities or two flow out causalities) [12]. Bond-Graphs based modeling being a graphical modeling technique provides a simplified procedure for developing state-space models of interacting physical systems [14]. The order of the state-space model is determined by counting the number of energy storing (*C* and *I*) elements in the Bond-Graph model. The derivation can either be carried out using old fashioned pen and paper or by using any of the software packages like 20-Sim© where all intermediate steps between developing a Bond-Graph representation of any dynamic physical system and arriving at its state-space model are done by the software and the mathematics remains in the background [15], [16]. This automated modeling approach reflects on a mechatronic system designer's job as he/she can now devote more time and effort towards handling the complicated nature and functional design aspects of the system without getting involved with lengthy derivations. Also software assisted modeling of physical systems using Bond-Graphs is object oriented. Object oriented modeling refers to physical system representation by various classes of reusable models, sub-models, sub sub-models... arranged in a hierarchical manner where the information pertaining to a particular model class is accessible only through variables associated with mathematical equations defining the class and identification of nature of causality on its proximities. These model classes when employed as structural elements, blocks or objects for constructing detailed models of physical systems carry their properties along. Such objects can be put together to develop designs for physical systems using a suitable tool for their manipulation [17], [18]. Figure 2 contains notation for all nine elements used in Bond-Graphs based physical systems modeling.

When using Bond-Graphs for mechatronic system representation, the mechatronic system can be treated as an *n*-port mechatronics network with e_l and f_l being system input effort and flow signals and e_n and f_n being system output effort and flow signals respectively as shown in Figure 3 [19].

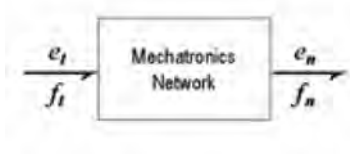

Fig. 3. A mechatronics network

Genetic programming

For developing computers (machines) that can think and act on their own with intelligence levels identical to human beings Alan Turing identified three basic types of search paradigms (logical, cultural and evolutionary search) [20], [21]. Here we are concerned with the evolutionary search approach which is modeled on Darwinian concepts of evolution and natural selection [22], [23].

Genetic programming invented by John R. Koza in 1990s is regarded as an extension of genetic algorithms attributed to John H. Holland [24], [25]. Both techniques are identical in nature except for representation of individuals which in case of genetic programming is parse trees based computer programs compared to fixed or variable length character strings in genetic algorithms [26], [27]. Representation is a major difference not only because it distinguishes the two techniques from each other but also because it greatly extends the problem handling capabilities of genetic programming. It is one of the most promising domain independent and object oriented evolutionary computation techniques [28], [29]. In genetic programming individuals indirectly represented as parse trees or genotypes with in the population are hierarchical compositions of functions and terminals pertaining to the type of the problem being solved. Functions may include arithmetic and logical operators and also custom user defined functions specific to the problem domain. Terminals are basically constants and inputs required for successful implementation of any problem. Search space contains all compositions of all available functions and terminals [30], [31].

Suitability of such compositions in providing a solution for the problem is evaluated through user defined fitness criterion. Mimicking evolution in nature genetic programming operating on a population of randomly generated individuals employs operators like reproduction, recombination or crossover, mutation, gene duplication, gene deletion and certain other mechanisms of developmental biology [32], [33]. In this simulated evolution scenario only the fittest off spring will be passed on to either join the next generation of individuals or become the solution at the end of the evolutionary search and synthesis process [34]. LISP has been one of the most

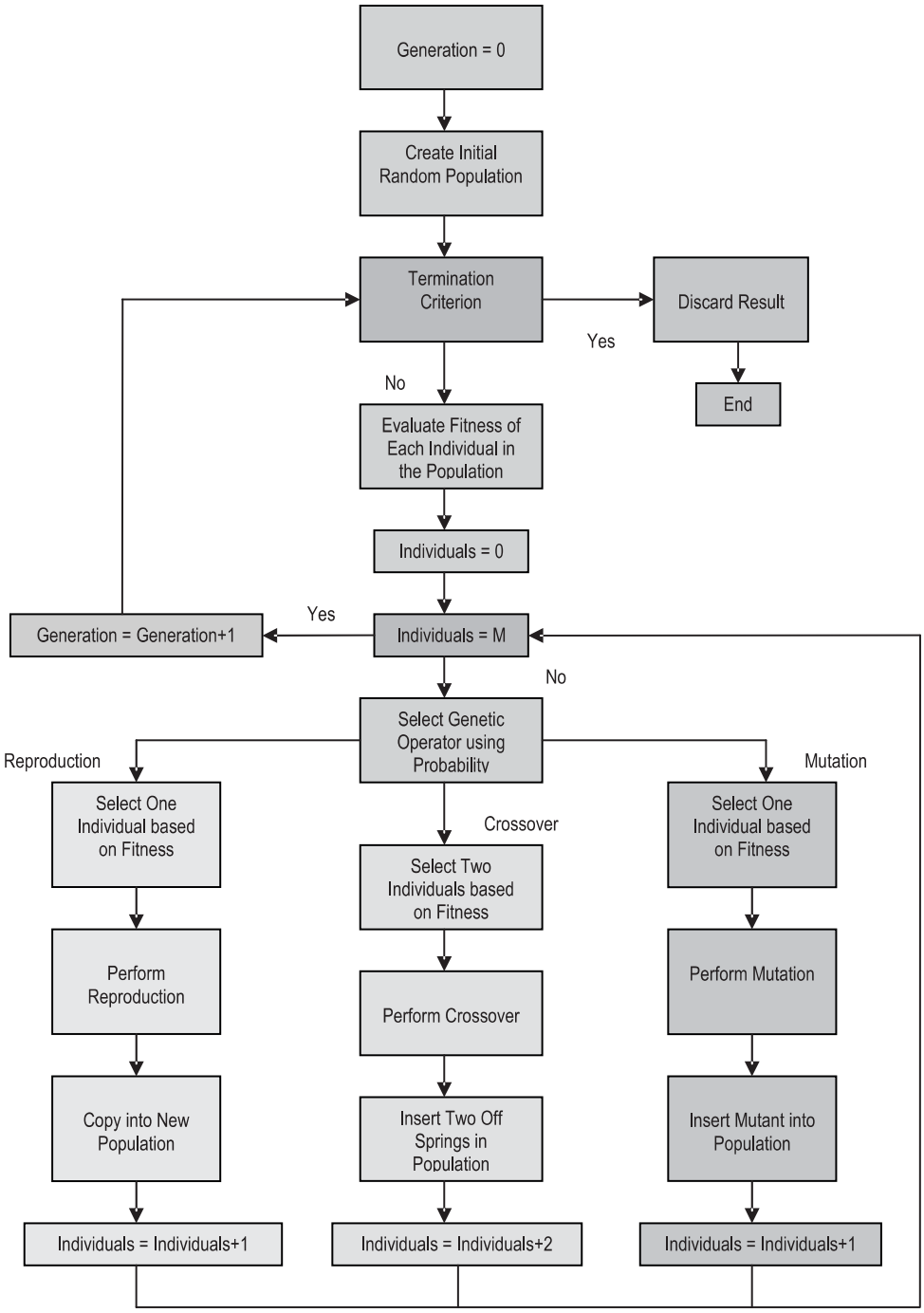

Fig. 4. Generalized representation of genetic programming operations

popular medium of implementing genetic programming problems. A generalized representation of genetic programming operations adapted from John R. Koza appears in Figure 4.

Automated design of mechatronic systems

By definition automated design is an essential part of genetic programming paradigm. Genetic programming receives a high level statement of a problem's requirements from the user and attempts to create a computer program that provides a solution for the problem. For a true automated and/or evolutionary invention machine it is desirable that the user provided information should be as minimal as possible and the topology (size and shape) of the evolved design should be part of the answer and not the question being supplied by the user [35].

For engineering design problems "the most critical issues are automation of the design process and use of a unified design tool" [36], [37]. There have been a number of research efforts aimed at exploring the combination of

genetic programming with physical systems modeling to find improved and even human compatible engineering designs. John R. Koza presents a single uniform approach using genetic programming for automatic synthesis of both the topology and sizing of a suite of various prototype analog circuits including low pass filters, operational amplifiers and controllers etc. His design approach requires a different simulation code or tool for each application [38]. "Writing a simulation code for an application is a very time consuming and tedious job. If the design applications or domains are different one must be able to write or link to a simulation code for each new application" [39], [40].

From our previous discussion we know that object oriented nature of Bond-Graph elements enables them to be manipulated using a suitable tool like genetic programming for developing statistically structured but indirectly (parse tree) represented Bond-Graph models of physical systems. Based on the idea of automatic programming when these Bond-Graph elements or objects are connected to generate models of physical systems the simulation code contained with in the object is automatically configured to make connections with other objects (observing principle of causality) while the continuity of the hidden mathematical model is preserved. Now the basic concept of automated design tends to replace the role of domain knowledge and reduce interaction between designer and design platform with the help of abundant computational resources available these days. "By combining unified modeling and analysis capabilities of Bond-Graphs with powerful topologically open ended synthesis and search capability of genetic programming a software system and methodology that is able to evolve innovative mechatronic (multi energy domain dynamic systems) design solutions represented as Bond-Graph models has been developed with ever improving performance in an iterative loop of synthesis, analysis and feedback to the design process" [41], [39]. Extensive development on this automated and unified design approach has been carried out at Genetic Algorithms Research and Applications Group of Michigan State University, East Lansing under the guidance of Erik D. Goodman and William F. Punch III since late 1990s. "Genetic programming based techniques are capable of synthesizing designs of arbitrary complexity as the representation of designs is entirely open ended. The

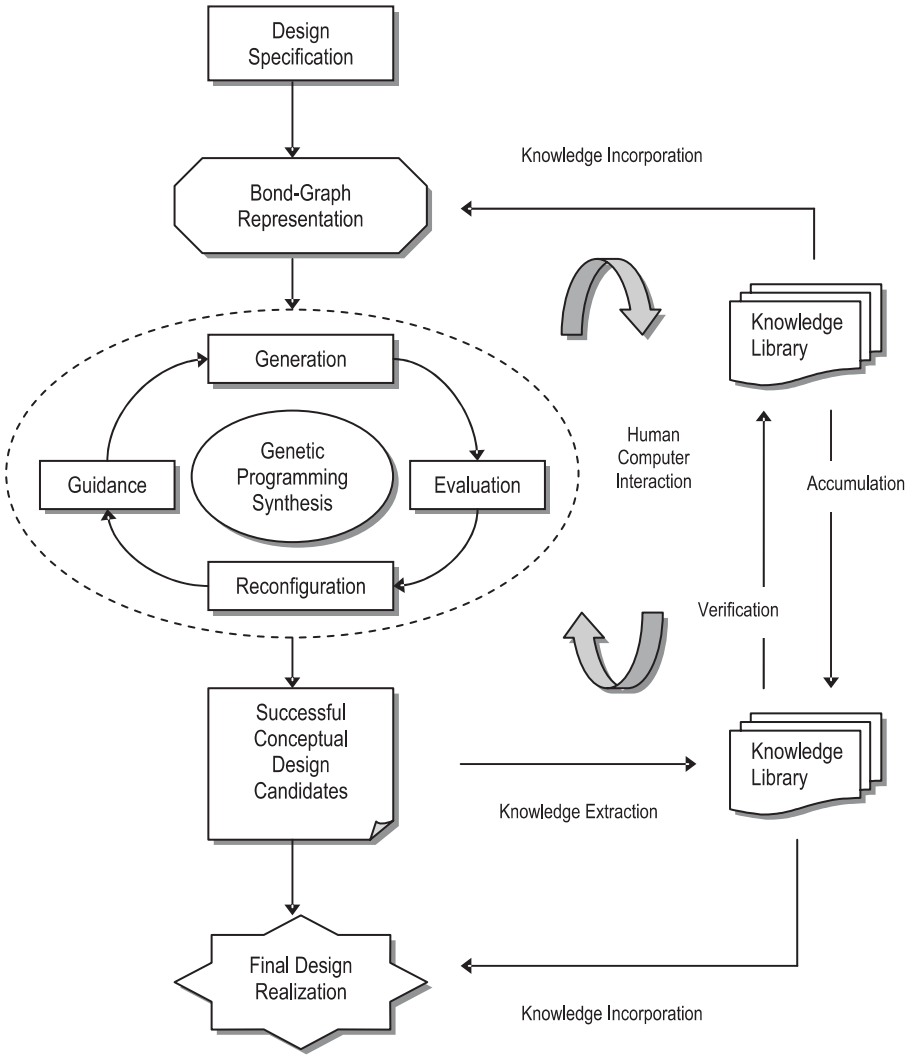

Fig. 5. A novel methodology for automated design of mechatronic systems

synthesis operators will allow the algorithm to combine building blocks of arbitrary size and shape from one design to insert into another based on what it has learned about the performance from Bond-Graph analysis and simulation of both designs [42]. This synthesis process is made more efficient because the Bond-Graph designs can be evaluated in two stages. The first determines whether the component connections in the candidate design satisfy certain rules of causality in their pattern of inter connection and this can be determined quickly using the researchers' existing Bond-Graph analysis tools. Then for designs which pass that criterion more expensive calculation of their performance in fulfilling the specified design objectives is done again using existing tools which the researchers need only adapt to this use" [39], [19], [43].

Figure 5 suggests an automated design approach for generating mechatronic systems identified by Wang et al. in 2005 [19]. According to this methodology for any mechatronic system a start up design in an embryonic state (represented as a Bond-Graph model) is specified at the initial design stage. Then using generative encoding its indirect representation (a genetic programming parse tree or genotype) is developed and transferred to the genetic programming software tool which evolves individuals with in the initial population using a set of basic construction functions, evaluates them according to the fitness criterion (a user specified function), reconfigures the population when required and repeats the process until the design criterion is satisfied. The process of decoding the genotype tree maps it into a phenotype which is an abstract topological description of the design of the mechatronic system using Bond-Graphs. Physical design realization is carried out to relate each abstract element of the Bond-Graph model to the corresponding components in various energy domains. During this whole process information is extensively exchanged with knowledge caches and incorporated into initial and final stages of the design process. The knowledge library serves as a dynamic database of domain knowledge that is constantly updated and the contents are accessible to designers (now effectively in supervisory role) to verify designs in an integrated automated design and information handling system. The input source to the knowledge library is the set of successful conceptual design candidates (outcome of the genetic programming based synthesis process) which means that only the best of all sorted information is stored for further reference and improvement [44], [45].

Automated synthesis of a primordial mechatronic system

In this section a design example is included which explains the implementation and functionality of the automated design methodology [44], [45]. Systematic procedure followed for implementing the said automated design methodology is outlined as follows: (a) An embryo Bond-Graph model is specified. (b) First population of genetic programming trees is created. (c) Each individual is evaluated for fitness using fitness function. (d) Genetic programming operations i.e. selection, reproduction, crossover and mutation are performed for each population. (e) Final design is realized if termination condition of genetic programming run is satisfied. (f) Otherwise the process is repeated starting from fitness evaluation of each individual. A two step process is employed for evaluation of evolved Bond-Graph models. First each model is analyzed for causality and then state equations are derived identifying whether the system is linear or not. In the second step fitness of each model is analyzed using fitness criterion.

Genetic Programming Studio 1.0 is employed in this application which is based on lil-gp 1.01 kernel but offers a visual platform for executing genetic programming code [46], [47]. The code has been written using MS Visual C++ 6.0. Six different types of code files have to be developed. 1. protoapp.h contains prototypes of functions that app.c includes. 2. appdef.h contains #defines of the application. 3. app.h contains global data and any other function defined by the user. 4. app.c contains software specific functions that help in input/output procedures. 5. function.h contains prototypes of functions and terminals of the problem. 6. function.c contains functions and terminals that are used for building the individual. Also files like epgdll.h, epgdll.c, defines.h, types.h, syscon.h and syscon.c are included for creating dll or dynamic linked library files. These files are software kernel files and are not to be modified. The genetic programming parameters are saved in the problem set file with the extension .epg. This software also offers a simulation tool which can be used for representing the genetic programming individuals or genotypes in LISP format. The code is compiled and the dll file is generated through MS Visual C++ 6.0. Two target eigen values $-1\pm 2j$ are selected represented by cross marks on complex plane as in Figure 6 and a Bond-Graph model of a physical system whose characteristic equation on solving for roots (poles) generates these eigen values is to be evolved.

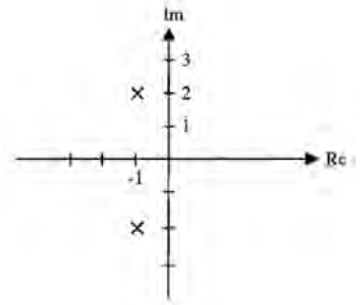

Fig. 6. Representation of $-1\pm 2j$ on complex plane

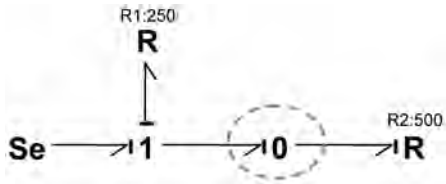

Fig. 7. The embryo Bond-Graph model

In Figure 7 an embryo Bond-Graph model is specified with only one modifiable site highlighted by a dashed oval

marking called the write head [36]. The fitness function includes two parameters namely raw fitness and normalized fitness. Raw fitness $Fitness_{Raw}$ is the sum of distances between target eigen values and the nearest solution eigen values after they have been paired. Normalized fitness $Fitness_{Norm}$ is calculated according to the relation given as equation 1.

$$Fitness_{Norm} = 0.5 + \frac{1}{1 + Fitness_{Raw}} \quad (1)$$

Table I. Function and terminal descriptions

Function	Description
f_tree	Generate a tree model
f_add_C	Add a C element to a junction
f_add_R	Add a R element to a junction
f_add_I	Add an I element to a junction
f_insert_J0	Insert a zero junction in a bond
f_insert_J1	Insert a one junction in a bond
replace_C	Replace with C element
replace_R	Replace with R element
replace_I	Replace with I element
f_add_ERC	Add two ERCs
f_del_ERC	Delete two ERCs
end_A	End terminal for add element
end_I	End terminal for insert element
end_R	End terminal for replace element
ERC	Ephemeral Random Constant

Table 2. Genetic programming parameters

Number of Generations	100-500
Population Size	100-2500
Initial Population	Half and Half
Sub Populations	10
Maximum Nodes	300
Initial Depth	3-6
Maximum Depth	17
Selection	Tournament
Size	7
Crossover	0.9
Mutation	0.1

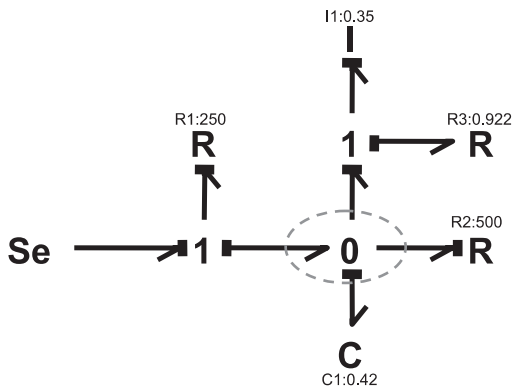

Fig. 8. The final simplified Bond-Graph model

The fitness function is determined by normalizing the relative distance based error into a fitness value between 0 and 1 through a scaling rule given as: if error/order < 0.1 then fitness = 0.1/(0.1 + error/order) otherwise it is taken as 5.05/(10 + error/order) where order is the number of energy storing elements in the Bond-Graph model.

A list of genetic programming functions and terminals along with their descriptions appears in Table.1. The function f_add_R requires an additional parameter called ephemeral random constant or ERC. It is a special terminal with a fixed value and when an ERC terminal is generated either during the filling of the initial population or by mutation later in the run, a value is attached to that terminal and remains unchanged by subsequent operations [47].

The genetic programming parameters used in the experiment have been included in Table.2. The software has been installed on a DELL/Pentium-III/1.0GHz and 256MB RAM personal computer running Windows XP/2002/SP-1. Three different random seeds were used and the experiment was repeated three times with population sizes of 100, 1000 and 2500 with different number of generations [48]. When different Bond-Graph functions can be applied to the same write head the technique is termed as strongly typed genetic programming. Add functions can only be applied to a junction while insert functions are only applied to a bond. Replace functions change the type of the Bond-Graph element and are node specific. Arithmetic functions of addition and subtraction are carried out by f_add_ERC and f_del_ERC respectively.

An illustration of the developmental/generative procedure followed for constructing a phenotype or Bond-Graph model (appearing on lower right corner) from a genotype or genetic programming tree (appearing on top left corner) is included as Figure 9 [36]. One genetic programming tree represents one individual. As mentioned earlier using LISP format of representing genetic programming trees the simulation tool of Genetic

Table 3. Summary of results

Target Eigen Values	
$-1\pm2j$	
Solution Eigen Values	
$-0.78\pm1.063j$	
Average Distance Error	
0.961	
Evolved Structure on Write Head	
R Elements	1
C Elements	1
I Elements	1
Junctions	1
Bonds	4
Bond-Graph Element Values	
R Element	0.922
C Element	0.42
I Element	0.35

Programming Studio 1.0 is employed to print long hand versions of such genotypes which are simplified and decoded (transformed into phenotypes or Bond-Graph models) for further processing like derivation of mathematical models and validation of the evolved design etc. as deemed necessary by the designer.

The average distance error e between target and solution eigen values is calculated using distance formula for two pairs of numbers (x_1, y_1) and (x_2, y_2) given as equation 2.

$$e = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2}$$

The best solution eigen values compared to target eigen values are included in Table.3 along with average distance error. This table also contains number of R, C, I and junction elements added to the write head. Numerical values of these one port elements are also shown. It is to be noted that eigen values are determined using the A matrix of the Bond-Graph model (when state-space equations are written in matrix form assuming the system is linear) containing state variables contributed by the energy storing C and I elements.

The evolved Bond-Graph model of the physical system in Figure 8 is analyzed using 20-Sim© modeling and simulation software [48]. The model contains two energy storing elements I_1 and C_1 therefore it is identified as a second order open loop system with two state variables. The evolved parameters are contained with in the periphery of the dotted square box. The general state-space representation for this Bond-Graph model is given as appearing in equations 3 and 4 [12].

$$\frac{d}{dt}\{X\} = A\{X\} + B\{U\} \tag{3}$$

$$\{Y\} = C\{X\} + D\{U\} \tag{4}$$

In equations 3 and 4 $\{X\}$ is vector of states (momentum P and displacement Q), n is number of states, A is $n \times n$ square matrix, B is $n \times m$ matrix (m is the number of sources), $\{U\}$ is vector of sources (Se and Sf), $\{Y\}$ is vector of observer states (outputs), l is number of observer outputs, C is $l \times n$ matrix and D is $l \times m$ matrix. The poles of the physical system being represented by this Bond-Graph model are determined by calculating eigen values from matrix A in equation 3 using relation $|A - \lambda I| = 0$ where I is

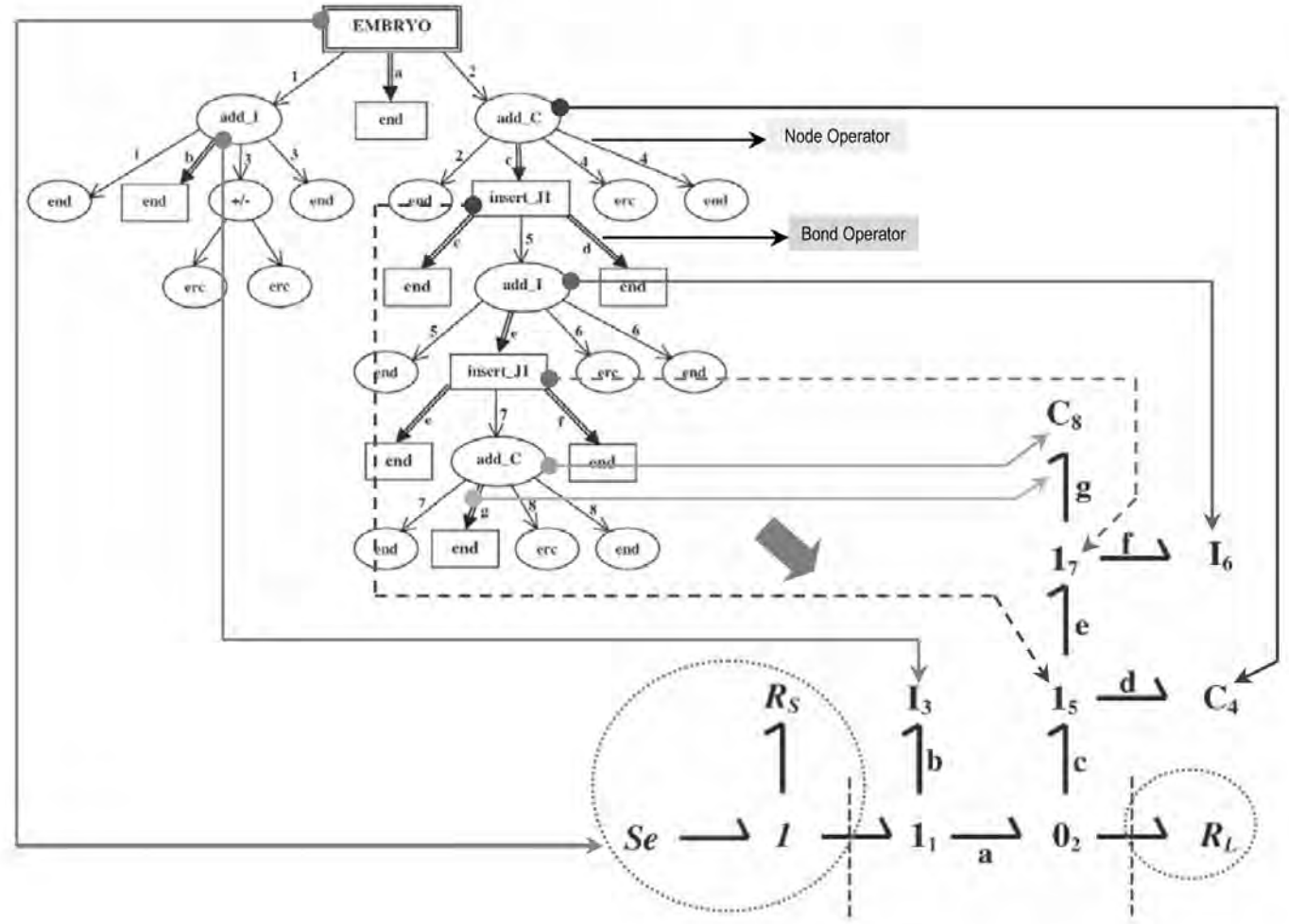

Fig. 9. Developmental/generative procedure for genotype-phenotype mapping

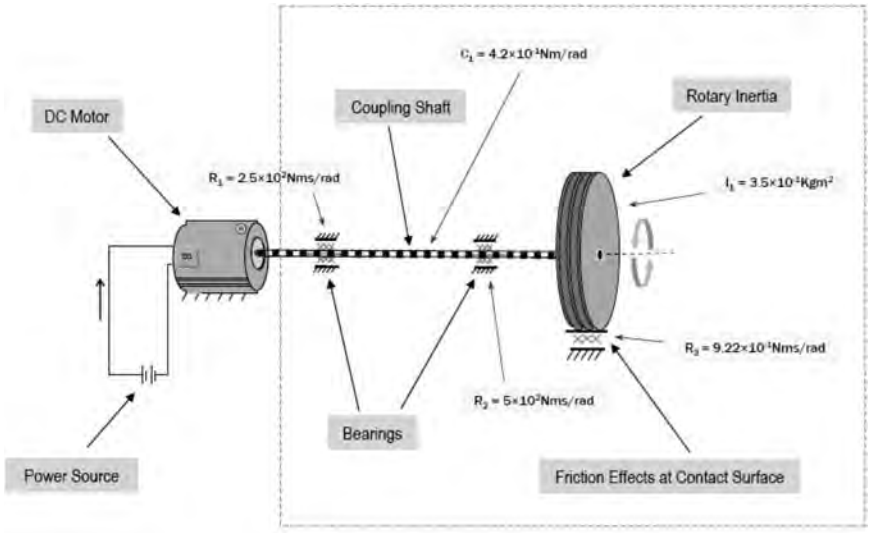

Fig. 10. Physical design realization of the evolved Bond-Graph model

identity matrix of order $n \times n$. In due course of the procedure followed for dynamic analysis of the system source of effort *Se* is replaced with a modulated source of effort *MSe* and a motion profile tool provided by the 20-Sim© modeling and simulation software is added to the workspace and connected to *MSe*. Motion profile selected is ramp with unit step as the input or excitation. Output signal is position or the observed output state is displacement *x(t)*. Values of input parameters are included in Table.4.

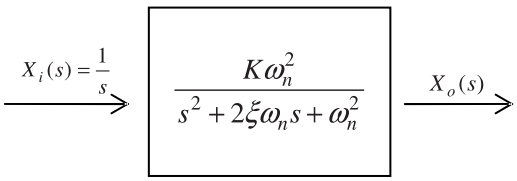

Fig. 11. Step response of a second order system with Damping Ratio $\xi < 1$

Table 4. Values of input parameters

Start Time	Rise Time	Stop Time	Amplitude
0 s	1 s	10 s	1

The response of the system to the unit step input is plotted in Figure 12. The values observed from the response curve appear as equations 5-8.

Settling Time = T_s = 5 s (5)
Rise Time = T_r = 1 s (6)
Peak Time = T_p = 2 s (7)
Damping Ratio = ξ = 0.591 (8)

For evolved eigen values -0.78 ± 1.063*j* the maximum average distance error turns out to be 0.961. The natural frequency ω_n, damped natural frequency ω_d and time period τ of the system are calculated using equations 9-12 [49].

$$T_s = \left[\frac{1}{\xi\omega_n}\right] \ln 50 \quad (9)$$

$$\tau = \frac{2\pi}{\omega_d} \quad (10)$$

$$\omega_n = \frac{2\pi}{\tau\sqrt{1-\xi^2}} \quad (11)$$

$$\omega_d = \omega_n\sqrt{1-\xi^2} \quad (12)$$

Substituting values of settling time and damping ratio from equations 5 and 8 respectively into equation 9 gives natural frequency of the system ω_n equal to 1.319 rad/s and damped natural frequency of the system thus becomes ω_d = 1.064 rad/s. Value of time period τ is calculated as 5.904 s/rad. Value of percent overshoot *%OS* = 10% is determined using equation 13 same as observed from the output curve in Figure 12 [50].

$$\%OS = e^{-\xi\pi/\sqrt{1-\xi^2}} \quad (13)$$

Relations for value of attenuation σ, rise time T_r, peak time T_p and maximum overshoot M_p appear as equations 14-18 [51].

$$\sigma = \xi\omega_n \quad (14)$$

$$T_r = \frac{\pi - \beta}{\omega_d} \quad (15)$$

$$\beta = \tan^{-1}\frac{\omega_d}{\sigma} \quad (16)$$

$$T_p = \frac{\pi}{\omega_d} \quad (17)$$

$$M_p = e^{-\left(\frac{\sigma}{\omega_d}\right)\pi} \quad (18)$$

Second order system characteristic response $X_o(s)$ and second order system transfer function $G(s)$ is given by equations 19 and 20 respectively [50].

$$X_o(s) = \frac{K\omega_n^2}{s(s^2 + 2\xi\omega_n s + \omega_n^2)} \tag{19}$$

$$G(s) = \frac{K\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{20}$$

Substituting values in equation 19 the system transfer function appears as in equation 21.

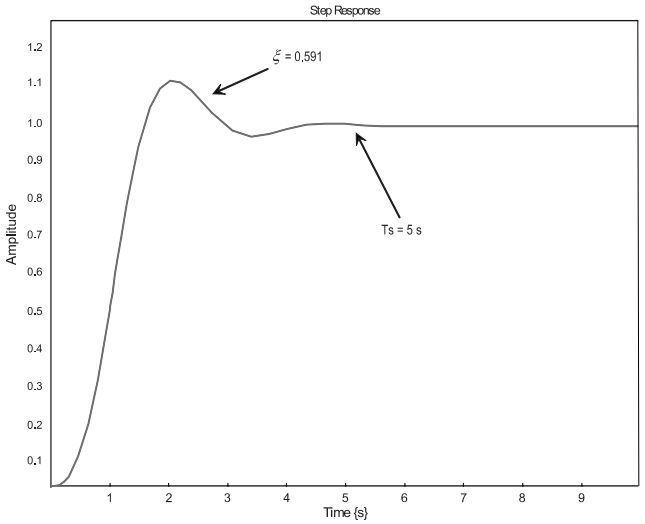

Fig. 12. System response to a unit step input

$$G(s) = \frac{(1.319)^2}{s^2 + 2(0.591)(1.319)s + (1.319)^2} \tag{21}$$

After simplification the system transfer function $G(s)$ is given by equation 22. Value of K or steady state gain is taken as unity.

$$G(s) = \frac{1.739}{s^2 + 1.56s + 1.739} \tag{22}$$

From equation 22 the characteristic equation for this particular system becomes $F(s) = s^2 + 1.56s + 1.739$.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{23}$$

$$x = \frac{-1.56 \pm \sqrt{(1.56)^2 - 4(1)(1.739)}}{2(1)} \tag{24}$$

$$x_{1,2} = -0.78 \pm 1.063j \tag{25}$$

Using quadratic formula roots of the system equation or poles of the physical system $s_{1,2} = -0.78 \pm 1.063j$ are determined as in equations 23 and 24 respectively. The rise time, settling time and damping ratio are typical of this type of systems. Using equations 14, 16 and 18 value of attenuation σ is determined as 0.77 rad/s, β is 54° and maximum overshoot M_p is 10.3%.

Conclusion

The methodology discussed in preceding sections has been proposed for unified and automated design of mechatronic or multi domain dynamic systems using Bond-Graphs for system representation and genetic programming for exploring the design space in an open ended but statistically structured manner [52]. The robustness of this novel unified/automated design approach towards system synthesis lies in compactness of the genetic programming code and fitness evaluation of the evolved designs. The complications associated with implementation of the methodology especially code development and final elucidation is one of the reasons that this research area remains relatively less explored. The full potential of this emerging automated design methodology can only be realized through development of an integrated design environment or IDE complete with a knowledge library and supporting features required for handling all elements of automated design including specification of Bond-Graph embryo, coding, genetic programming based search/ synthesis and mapping of genotypes into phenotypes along with an in-package tool to simplify evolved Bond-Graph models. Physical design realization and testing can be carried out using other well established design and analysis platforms like 20-Sim©.

Surely the power of Bond-Graphs in representing physical systems and genetic programming in exploring open ended design space is revealed when these two elegant techniques come together for automated design of mechatronic or multi domain dynamic systems. The need of an integrated design environment or IDE that makes both of these techniques available in one package should stimulate efforts among international research groups for development of such a software which will definitely serve as a useful tool in hands of engineers and designers working in this exciting research area. It will also make this intelligent automated design system incorporating Turing's concept of evolutionary search for machine intelligence available for relatively complicated systems without having to deal with linking two or more software packages for implementing an automated design problem. As described earlier the powerful multi energy domain modeling features offered by Bond-Graphs come in handy to develop detailed dynamic representations of natural systems. Such a software can also serve as a basis for addressing and validating reservations about the strength of evolutionary search and state of available knowledge and technology when it comes to design of physical systems (appearing in form of Bond-Graph models) as complex as found in nature. An observation pertaining to Bond-Graphs based representation is the limitation imposed due to lack of two port elements transformer *TF* and gyrator *GY* on the automated synthesis process which tends to restrict the evolutionary search and synthesis to one particular energy domain at any time without scaling. Direct transition from one energy domain to the other can be made if a genetic programming function for gyrator element is available thus

extending the range of the design approach to nearly complete multi energy domain systems.

Acknowledgment

The authors are indebted to College of Electrical and Mechanical Engineering, National University of Sciences and Technology, Islamabad, Pakistan for continued support and provision of research facilities without which this work would not have been possible.

REFERENCES

1. Rolf Isermann, "Modeling and design methodology for mechatronic systems", *IEEE/ASME Transactions on Mechatronics*, Vol. 1, No. 1, March 1996, pp. 16-28.
2. Job van Amerongen and Peter C. Breedveld, "Modeling of physical systems for the design and control of mechatronic systems", *Annual reviews in Control*, Vol. 27, 2003, pp. 87-117.
3. Michael B. Histand and David G. Alciatore, "Introduction", *Mechatronics and Measurement Systems*, New York: McGraw Hill, 1998.
4. Kamal Y. Toumi, "Modeling, design and control integration: A necessary step in mechatronics", *IEEE/ASME Transactions on Mechatronics*, Vol. 1, No. 1, March 1996, pp. 29-38.
5. Henry M. Paynter, "An epistemic prehistory of Bond-Graphs", *Bond-Graphs for Engineers: IMACS Transactions Series*, Peter C. Breedveld and G. Dauphin-Tanguy, eds., *Elsevier*, Amsterdam, 1992, pp. 3-17.
6. Henry M. Paynter, *Analysis and Design of Engineering Systems*, Reading: MIT Press, 1961.
7. R. W. Newcomb, *Linear Multi Port Synthesis*, New York: McGraw Hill, 1966.
8. Peter J. Gawthrop and L. Lorcan Smith, *Metamodeling: Bond-Graphs and Dynamic Systems*, New York: Prentice Hall, 1996.
9. Peter C. Breedveld, "Thermodynamic Bond-Graphs and the problem of thermal inertness", *Journal of Franklin Institute*, Vol. 314, No. 1, 1982, pp. 15-40.
10. Dean C. Karnopp, "Retaining analog intuition in a digital world with Bond-Graphs", *Mathematics and Computers in Simulation*, Vol. 53, 2000, pp. 219-226.
11. Peter C. Breedveld, "Port based modeling of mechatronic systems", *Mathematics and Computers in Simulation*, Vol. 66, 2004, pp. 99-127.
12. Dean C. Karnopp, Donald L. Margolis, and Ronald C. Rosenberg, *System Dynamics: Modeling and Simulation of Mechatronic Systems*, New York: John Wiley, 2000.
13. Amalendu Mukherjee and Ranjit Karmakar, *Modeling and Simulation of Engineering Systems through Bond-Graphs*, New Delhi: Narosa, 2000.
14. Peter J. Gawthrop and Geraint P. Bevan, "Bond-Graph modeling", *IEEE Control Systems Magazine*, Vol. 27, No. 2, April 2007, pp. 24-45.
15. Dean C. Karnopp and Donald L. Margolis, "The language of interaction", *American Society of Mechanical Engineers*, January 2001, pp. 1-4.
16. 20-Sim©, Version 3.3, Control Lab. Products B.V., PO Box 217, 7522 NB Enschede, The Netherlands.
17. Wolfgang Borutzky, "Relations between Bond-Graph based and object-oriented physical systems modeling", *Proceedings of International Conference on Bond-Graph Modeling and Simulation,* San Francisco, January 17-20, 1999, pp. 11-17.
18. Saheeb A. Kayani, "On automated design of mechatronic systems through Bond-Graphs and genetic programming", *IEEE Multi Disciplinary Engineering Education Magazine*, Vol. 2, No. 4, December 2007, pp. 15-17.
19. Jiachuan Wang, Zhun Fan, Janis P. Terpenny, and Erik D. Goodman, "Knowledge interaction with genetic programming in mechatronic systems design using Bond-Graphs", *IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews*, Vol. 35, No. 2, May 2005, pp. 172-182.
20. Alan M. Turing, "Computing machinery and intelligence", *Mind*, Vol. 59, No. 236, October 1950, pp. 433-460.
21. John R. Koza, Forest H. Bennet III, David Andre, and Martin A. Keane, "Genetic programming: Turing's third way to achieve machine intelligence", *Evolutionary Algorithms in Engineering and Computer Science*, Kaisa Miettinen, et al., eds., Jyvaskyld, 1999, pp. 185-197.
22. Kevin Padian, "Darwin's enduring legacy", *Nature*, Vol. 451, No. 7179, February 2008, pp. 632-634.
23. John R. Koza, Forrest H. Bennet III, David Andre, and Martin A. Keane, *Genetic Programming III: Darwinian Invention and Problem Solving*, San Francisco: Morgan Kaufmann, 1999.
24. John H. Holland, *Adaptation in Natural and Artificial Systems*, 2nd ed., Cambridge: MIT Press, 1992.
25. John R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge: MIT Press, 1992.
26. David E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, New York: Addison Wesley, 1989.
27. Melanie Mitchell, *An Introduction to Genetic Algorithms*, Cambridge: MIT Press, 1996.
28. Haym Hirsh, Wolfgang Banzhaf, John R. Koza, Conor Ryan, Lee Spector and Christian Jacob, "Genetic programming", *IEEE Intelligent Systems*, Vol. 15, No. 3, 2000, pp. 74-84.
29. John R. Koza, Martin A. Keane, and Matthew J. Streeter, "What's AI done for me lately? - Genetic programming's human competitive results", *IEEE Intelligent Systems*, Vol. 18, No. 3, 2003, pp. 25-31.
30. John R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, Cambridge: MIT Press, 1994.
31. William B. Langdon and Riccardo Poli, *Foundations of Genetic Programming*, Berlin: Springer-Verlag, 2002.
32. John R. Koza, "Genetic evolution and co-evolution of computer programs", *Artificial Life II*, Christopher Taylor Langton, et al., eds., New Mexico: Addison Wesley, 1991, pp. 603-629.
33. John R. Koza, "The genetic programming paradigm: genetically breeding populations of computer

programs to solve problems", *Dynamic, Genetic and Chaotic Programming*, Branko Soucek, et al., eds., New York: John Wiley, 1992, pp. 203-321.
34. L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*, New York: John Wiley, 1966.
35. John R. Koza, "Human competitive machine intelligence by means of genetic programming", *IEEE Intelligent Systems*, Vol. 15, No. 3, 2000, pp. 76-78.
36. Kisung Seo, Zhun Fan, Jianjun Hu, Erik D. Goodman, and Ronald C. Rosenberg, "Towards a unified and automated design methodology for multi domain dynamic systems using Bond-Graphs and genetic programming", *Mechatronics*, Vol. 13, 2003, pp. 851-885.
37. Zhun Fan, Jianjun Hu, and Erik D. Goodman, "Exploring open ended design space of mechatronic systems", *International Journal of Advanced Robotic Systems*, Vol. 1, No. 4, 2004, pp. 295-302.
38. John R. Koza, Forrest H. Bennett III, Jason Lohn, Frank Dunlap, Martin A. Keane, and David Andre, "Automated synthesis of computational circuits using genetic programming", *Proceedings of IEEE International Conference on Evolutionary Computation*, Indianapolis, 1997, pp. 447-452.
39. Erik D. Goodman, Kisung Seo, Ronald C. Rosenberg, Zhun Fan, Jianjun Hu, and B. Zhang, "Automated design methodology for mechatronic systems using Bond-Graphs and genetic programming", *Proceedings of NSF Design, Service and Manufacturing Grantees and Research Conference*, San Juan, January 2002, pp. 206-221.
40. Jiachuan Wang and Janis P. Terpenny, "Integrated active and passive mechatronic system design using Bond-Graphs and genetic programming", *Proceedings of Genetic and Evolutionary Computation Conference*, Chicago, July 2003, pp. 322-329.
41. Zhun Fan, Kisung Seo, Ronald C. Rosenberg, Jianjun Hu and Erik D. Goodman, "Computational synthesis of multi domain systems", *Proceedings of AAAI Spring Symposium on Computational Synthesis,* Stanford, March 24-26, 2003, pp. 59-66.
42. B. Danielson, J. Foster, and D. Frincke, "GABSys: using genetic algorithms to breed a combustion engine", *Proceedings of IEEE International Conference on Evolutionary Computation,* Anchorage, May 4-9, 1998, pp. 259-264.
43. Jianjun Hu and Erik D. Goodman, "Topological synthesis of robust dynamic systems by sustainable genetic programming", *Genetic Programming: Theory and Practice II*, Una-May O'Reilly et al., eds., Boston: Springer, 2005, pp. 143-157.
44. Saheeb A. Kayani and Afzaal M. Malik, "Automated design of mechatronic systems using Bond-Graph modeling and simulation and genetic programming", *Proceedings of IEEE International Bhurban Conference on Applied Sciences and Technology*, Islamabad, January 8-11, 2007, pp. 104-110.
45. Saheeb A. Kayani and Afzaal M. Malik, "Combining Bond-Graphs with genetic programming for unified/automated design of mechatronic or multi domain dynamic systems", *Proceedings of Genetic and Evolutionary Computation Conference*, London, July 7-11, 2007, pp. 2515-2518.
46. Andres del Campo Novales, Genetic Programming Studio 1.0, University of Cordoba, Spain, 1998.
47. Douglas Zongker and Bill Punch, *lil-gp 1.01 User Manual*, Michigan: Michigan State University, USA, 1996.
48. Saheeb A. Kayani, "Automated Design of Mechatronic Systems Using Bond-Graphs and Genetic Programming", Masters Research Thesis, Technical Library, College of E&ME/NUST, Rawalpindi, Accn. No. TH-386, March 2007, pp. 95-137.
49. Roland S. Burns, *Advanced Control Engineering*, Oxford: Butterworth-Heinemann, 2001.
50. Katsuhiko Ogata, *Modern Control Engineering*, New Jersey: Prentice Hall, 1997.
51. William J. Palm III, *Control Systems Engineering*, New York: John Wiley, 1986.
52. William B. Langdon, *Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming*, Amsterdam: Kluwer, 1998.