NEW DEVELOPMENTS IN BOND GRAPH MODELING SOFTWARE TOOLS:
THE COMPUTER AIDED MODELING PROGRAM CAMP-G AND MATLAB

Jose J. Granda
Professor
Department of Mechanical Engineering
California State University, Sacramento
Sacramento, Ca 95819, USA
email: grandajj@ecs.csus.edu
Phone: (916)278-5711  Fax:(916) 278-5949

Jim Reus
Lawrence Livermore National Laboratory
Livermore, California  94550
email:  reus@llnl.gov

ABSTRACT

The Computer Aided Modeling Program (CAMP-G) is a Bond Graph Modeling tool designed to make physical system models in graphical form and to generate systems of equations in source code form so that computer simulation programs and now MATLAB can analyze dynamic systems. Current technology allows the creation of system models using CAMP-G and performing simulations using general purpose simulation languages such as ACSL, DSL, CSSL or the user’s own program. This paper investigates the ability to use Bond Graph Modeling technology with MATLAB and its tool boxes, a package oriented to matrix state variable formulation and control system design. Basic principles of causality and equation generation are presented in order to establish the theoretical basis for the logic. The analysis of multienery physical systems and control systems joins applications by block diagrams and bond graphs in a single model. The capabilities of CAMP-G to generate automatically source code models have been incorporated together with the mathematical capabilities of MATLAB. The new modeling environment created with this combination and the possible applications in system dynamics and control such as electro-mechanical and Mechatronics systems are discussed.

1. INTRODUCTION

Bond Graph modeling technology finds one of its most important applications in the analysis of dynamics systems that have components in different energy domains. Nowadays, the need of electro-mechanical system design dominates many applications in electrical and mechanical engineering. Other systems combine mechanical, hydraulic or pneumatic controls, thermal energy or a combination of all of these. Developments in the area of Mechatronics make bond graph modeling a trend for making models of physical systems for this type of multienergy systems. The idea of transforming a physical system bond graph model into source equations of motion was originally proposed by Granda [1]. It has become an established and well known simulation approach in the field of bond graph modeling. What is new here is that it is possible to create a bond graph model on the screen in graphical form using CAMP-G and produce MATLAB source code files for any systems represented by a bond graph model. The idea is to preprocess the bond graph and assist the user in the creation of models. If a change in the model is desired, editing the graphical representation of the model results in a new derivation of the system matrices in symbolic state variable first order form. Using MATLAB and its tool boxes it is possible then to perform simulations of complex non-linear dynamic systems and design control systems. All of this without the need of intermediate steps such as making a block diagram to describe the system equations in logical form. Normally the process requires the user to make a block diagram or derive equations of motion to describe them to a simulation language or to have on hand the system matrices or a transfer function before he uses MATLAB. The proposed approach is tailored to help the user generate these equations, generate the matrices and the transfer functions in the early stages of making the model.

Nise [2]. summarizes the conventional approach to six steps. 1. Determination of physical system and specifications from the requirements 2. Draw a functional block diagram. 3. Transform the physical system into an schematic. 4. Use schematic and obtain a mathematical model, a block diagram or state representation. 5. Reduce the block diagram to close loop system. 6. Analyze design and test. The proposed approach here is: 1. Identify physical system elements and specifications from the requirements (Word bond graph). 2. Draw a bond graph model using CAMP-G. 3. Obtain computer generated state variable matrix form and use MATLAB. 4. Analyze design and test.

Bond graphs can generate equations for the physical system, the “plant”, block diagrams can be used to describe additional logic for control systems. Here we concentrate on the “plant”, the physical model. The reason that this works is that the theoretical basis of bond graphs follows energy conservation methods that describe the physical system following the laws of physics and produce close loop forms of system equations in state variable form. Karnopp, Margolis, Rosenberg [3] describe this in detail. Block diagrams that describe a control system link up well with the individual internal signals (efforts and flows) or variables derived from bond graphs.

This paper proposes a combination of CAMP-G/ MATLAB as a major step in the design of multi energy dynamic systems. The feasibility of using the combination of CAMP-G, MATLAB and its tool boxes makes it possible to give the user a more physical understanding of the system by using bond graphs as the starting modeling method, making it automatic the derivation of system equations and MATLAB M files and finally making it easier to use tools that are of common knowledge to most systems and control engineers.

2. BASIC PRINCIPLES AND MATLAB FORMULATION

Let’s examine the basic theory and approach that MATLAB uses in order to solve differential equations in expressions or as matrices. MATLAB uses the state variable formulation. This means that a
second order system for example, needs to be described as a set of two first order differential equations. Once this form has been achieved and described in MATLAB, it is possible to use MATLAB tools for simulation or control system design. For example, let’s take the second order system equivalent in mechanical and electrical domains.

\[ i = \frac{Q}{C} \]

\[ V(t) = iR \]

\[ \frac{dQ}{dt} = \frac{1}{C}i(t) \]

\[ \frac{d^2Q}{dt^2} = \frac{1}{LC} \frac{dv}{dt} - \frac{1}{LC} VRQL \]

\[ \frac{dx}{dt} = u_2 \]

\[ \frac{dv}{dt} = -\frac{k}{m}x - \frac{b}{m}v - \frac{F(t)}{m} \]

The purpose in presenting both of these formulations is to establish the relation between the format of equations for different energy domains. Such format defines the system matrices, state variable vectors, inputs and derivatives. This state variable formulation can be entered in MATLAB in the form of a function described in an “.M” file. The standard procedure is to define a set of statements in and M file which are used to drive the computation of system matrices, perform the integration and prepare a set of output vectors which can be plotted to observe the results. The following MATLAB M file function u_prime defines the above state space equations (9).

\[ \text{function } u\_prime = \text{oscillat}(t,u) \]

\[ \text{global b m k F} \]

\[ u\_prime(1) = u(2); \]

\[ u\_prime(2) = -(k/m)*u(1)+(-b/m)*u(2)+(1/m)*F; \]

How does this form relate to bond graphs is the key in linking MATLAB to bond graph modeling and thus we discuss it next.

3. THE BOND GRAPH MODELING APPROACH

The formulation proposed by Granda [4] is used here as a basis for the creation of a computer generated set of MATLAB M files starting from a Bond Graph Model of the system. In order to illustrate this, let’s look at the example proposed above and solve it using CAMP-G and MATLAB. The simple bond graph shown below represents both the mechanical and electrical system.

For the mechanical system the substitution of variables using \( u_1 = x \) (displacement) and \( u_2 = v \) (velocity) is:

\[ \frac{dx}{dt} = u_2 \]

\[ \frac{dv}{dt} = \frac{1}{m} \left( \frac{d^2x}{dt^2} - \frac{k}{m}x - \frac{b}{m}v - \frac{F(t)}{m} \right) \]

\[ \frac{dx}{dt} = u_2 \]

\[ \frac{dv}{dt} = \frac{1}{m} \left( \frac{d^2x}{dt^2} - \frac{k}{m}x - \frac{b}{m}v - \frac{F(t)}{m} \right) \]

Granda [1], [4] proposes to follow the causality marks and write an algebraic expression for each effort and for each flow. One can start with any bond or do it sequentially, it makes no difference in generating a close loop set of equations. These forms a set of equations that defines not only every internal signal within the system but also calculate the first order derivatives, thus generating a state variable representation. This basic principle added to the fact that a computer program calculates the expressions on the left side of the equal sign, defines a set of output vectors (efforts and flows) at every step of integration. Also a vector of derivatives which is integrated by MATLAB using its differential equations solvers. A general purpose format is proposed here in order to
establish a basic structure to be used in automatic computer generated MATLAB. M files in source code form.

Before that is done it is necessary to address two issues that are new in this paper. First, the proposal that writing an orderly set of individual element equations and junction equations describes state variable format from a bond graph as proposed in [1]. This means if we follow the causality and write individual equations we will end up with a close loop set in state variable form. Such concept differs from the conventional method of obtaining system of equations from bond graphs such a outlined in Karnopp, Margolis, Rosenberg [3] where variable substitution is used at each step in the derivation in order to express the equations in terms of the states and the inputs. The second issue is whether this kind of equations produced as outlined in [1],[3] or in [4] will be suitable for MATLAB. The answer to this question is no. The reason is that MATLAB is not a simulation language such as ACSL (Advanced Continuous Simulation Language), but it is oriented more as a programming language. For this reason, the definition of variables in logical form is crucial to MATLAB. That being the case, changes all rules proposed in the approach that CAMP-G uses with ACSL. The translator in ACSL is the intelligent logical sorting of all expressions and blocks which allows an internal code generation and compilation so that an executable program calculates the simulation results. This action indicates an important feature in simulation languages which is different from MATLAB. This issue is crucial because without such logic, MATLAB will not calculate anything and we are stopped cold at start because the equations generated from a bond graph as in [1] and [3] do not follow logical sequence.

Two sets of equations are shown in Fig 3, which correspond to those discussed above. The CAMP-G generated equations for a simulation language such as ACSL and the new logically ordered set for MATLAB.

\[
\begin{align*}
\text{MATLAB type of equations} & \\
\text{ACSL type of code.} & \\
\end{align*}
\]

(a) \hspace{2cm} (b)

![Fig 3. CAMP-G generated control and differential equation files](image-url)

4. SYSTEM MATRICES GENERATION

CAMP-G will generate the A,B,C,D matrices in source code symbolic form. It uses the row vectors defining each state equation and symbolically generates the rows or the A and B matrices simultaneously. When the output vectors are examined also and the C and D matrices are generated. The computer generated code looks like the file below. Some specific plotting statements have been added.

CAMPQ.M

This M file acts as a main program that calls the differential equation solver ODE23 in MATLAB, Etter [7]. Others such as ODE45 can also be used. Numerical values have been added to the computer generated files.

**Fig 4. CAMP-G generated control and differential equation files**

CAMPQ.M

function p.qdot = bgequa(t,p.q) 
% global i2 r3 c4 se1 
% System Physical Parameters ........
% i2 = 0.2; r3 = 0.03; c4 = 1/1.96; 
% External inputs
% se1=0;
% Initial conditions vector ........
% p2in= 0 ; q4in=0.04 ;
% initial = [p2in q4in];
% Simulation Time Control
% t0=0; t=5;
% Solution of system equations 2-3 order R-T
% [f(p,q) = ode23(bgequa,t0,t,initial)];
% Plotting Results
figure(1)
subplots(211),plot(t,p_q(:,1),'m'),grid
subplot(212),plot(t,p_q(:,2),'b'),grid
figure(1)
% Plotting Results
% ...... Solution of system equations 2-3 order R-T
% [p2,q4] = ode23(bgequa,t0,t,initial);
% ...... Define System Variables ........
% p2= p.q(1);
% q4= p.q(2);
% ...... Define Derivatives and output variables ........
e1 = se1; 
e2 = p2/2; 
e3 = q4/c4; 
f1 = f2; 
f4 = f2; 
dp2 = dp2; 
dq4 = dq4; 
% ...... Build vector of derivatives
p.qdot = [dp2 dq4];

CAMPQSYM.M

% Generate State Space Symbolic formulation
% System Matrices A, B, C, D using bond graph equations
% Define input vector u
u(1,:)=se1; 
u(2,:)=0;
% Generation of System matrices

CAMPQ.M

% Define external inputs
i2 = 0.2; r3 = 0.03; c4 = 1/1.96;
% Define initial conditions
se1=0;
% Define simulation time
t0=0; t=5;
% Define system equations
[p2,q4] = ode23(bgequa,t0,t,initial);
% Plotting results
figure(1)
subplot(211),plot(t,p_q(:,1),'m'),grid
subplot(212),plot(t,p_q(:,2),'b'),grid
figure(1)
% Plotting Results
% ...... Solution of system equations 2-3 order R-T
% [p2,q4] = ode23(bgequa,t0,t,initial);
% ...... Define System Variables ........
% p2= p.q(1);
% q4= p.q(2);
% ...... Define Derivatives and output variables ........
e1 = se1; 
e2 = p2/2; 
e3 = q4/c4; 
f1 = f2; 
f4 = f2; 
dp2 = dp2; 
dq4 = dq4; 
% ...... Build vector of derivatives
p.qdot = [dp2 dq4];

CAMPQSYM.M

% Generate State Space Symbolic formulation
% System Matrices A, B, C, D using bond graph equations
% Define input vector u
u(1,:)=se1; 
u(2,:)=0;
% Generation of System matrices
5. GENERATION OF TRANSFER FUNCTIONS

Once these output vectors and matrices have been defined, all MATLAB commands and its toolboxes are available. It is possible to obtain the transfer functions numerically. Such is achieved by using the MATLAB ODE23 integration function in CAMPGEQ.M. The numeric transfer functions (Fig 7) for the frequency response plots shown in Fig 9. The vector of transfer functions for \( e3(s)/F(s) \), \( f4(s)/F(s) \) and \( \theta(s)/F(s) \) was generated in MATLAB by this procedure. These transfer functions correspond to output variables in relation to the input SE1(s). The output equations are generated in CAMP-G in terms of the state variables and the inputs, thus defining the C matrix. Now that the symbolic representation of the system matrices has been established it opens the door for the MATLAB Symbolic Toolbox or the MATHEMATICA capabilities to generate transfer functions in symbolic form, a very useful feature used in design of compensators for control systems. Chow [6], Cavallo [8]. The symbolic transfer functions generated this way corresponding to \( e3, \theta \) and the position angle \( \theta \) are shown in Fig 8. 

6. MULTIENERGY SYSTEMS

Bond Graph modeling is the appropriate method when systems have components in different energy domains but they act as a complete dynamic system. One of the most classic ones is the model of a DC motor. This is an electromechanical system. It was deliberately chosen because keeping continuity with the theory expressed above, it is important to link the approach to multi energy systems and demonstrate the automated computer formulation approach considering the electrical components and the mechanical ones as a single system. Nowadays the trend is to prepare mechanical engineers in electronics. The new area of Mechatronics is a perfect place for application of the techniques outlined here. Nise [2] analyzes the process for a d-c motor. A comparison of the approach is useful to illustrate the differences in the conventional approach outlined in Nise [2] and the one proposed here. Figure 10 shows the schematic of a d-c motor.
Such detection is a property of bond graph modeling methods that translates the schematic directly, a modeling problem is detected. The bond graph model will detect derivative causality. The solution is to join both inertia elements, the bond graph junction with a gyration which predict this even before any derivation of the equations has been attempted. The CAMP-G Bond Graph model shown in Fig 11. It is obvious in this example that there are two masses involved in the rotational motion, the armature and the load. Since they are coupled together, they are not independent motions. If we disregard this at start and model it as two separate inertia elements, the bond graph model will detect derivative causality. The solution is to join both inertial effects. Most of the literature about motors reveals the calculation of the equivalent inertia, but in most cases there is no explanation as to why this is necessary. The bond graph model makes this obvious. What is significant here is that if one translates the schematic directly, a modeling problem is detected. Such detection is a property of bond graph modeling methods which predict this even before any derivation of the equations has been attempted.

CAMP-G generated the .M input files which describes the equation of motion in first order form and with the momentum as the state variable. Speed control can be easily analyzed by tracking the velocity output signal in any of the bonds 8, 9 or 10. The position however needs another integration of the velocity. This can be accomplished in three ways. One is a dummy C element to produce the integration. Two is to increase the number of states in the MATLAB system equations and include the angular velocity as one of the states so that the position angle is calculated as a result of the integration of all state derivatives. The third way is to wait until the bond graph model is converted to the S domain in MATLAB and be treated as the “plant” and then add a (1/s) term to produce the integration. Incrementing the state space modifies the system matrices and in this case introduces a zero eigenvalue. Here, option two was chosen and the state space was increased to include the angular velocity f9 as one of the derivatives so that integration will produce the angle theta. Due to space limitations, one has to be very careful in defining the C matrix in order to define the position angle. The other one corresponds to the angular momentum. The MATLAB simulation results obtained by the integration of the differential equations generated in logical sequence and integrated within MATLAB. Fig 14 is generated by the use of the step input on the transfer function generated from the symbolic A,B,C,D matrices generated from the bond graph. The figure on the bottom of Fig 14 is exactly that of the top of Fig 13 for the position angle. The other one corresponds to the angular velocity of the armature and load (f9) shown at the bottom of Fig 13 and Fig 14 is exactly that of the top of Fig 13 for the position angle. The other one corresponds to the angular velocity of the armature and load (f9) shown at the bottom of Fig 13.
7. CONCLUSIONS

A new approach to generate computer models of physical and control systems has been presented here. The application of the method is oriented towards mechatronic systems. The area of electro-mechanical systems with intelligent electronic controls (Mechatronics) may find its most useful applications. Granda, Dauphin-Tanguy, and Rombaud [5] illustrate this. The combination of CAMP-G and MATLAB is a new tool useful in generating symbolic equations of motion and symbolic system matrices and symbolic transfer functions. It opens the door for all tool boxes MATLAB has for design of control systems. The transfer functions being numeric or symbolic, are the tools design engineers need to get insight understanding of a physical system.

One of the logical applications of this research is the simulation and analysis of systems that require block diagrams such as control systems and differential equations of physical dynamic systems. A control system model needs the description of the physical system, the “plant”. Since bond graphs provide a topological description and a direct transformation of reality into a computer model, makes them an ideal tool to generate the model from scratch. Transforming a bond graph model in the form of transfer functions in the S domain, allows building close loop feedback systems in the S domain. The transformation of the bond graph model “plant” into the S domain as outlined here, opens the door for a mixture of bond graphs and block diagrams in the S domain, the most common approach in the design of controls. A major principle of this approach in the fact that if the physical model is changed, the derivation of the equations and MATLAB M files in source code form can be generated quickly in CAMP-G. This can save a tremendous amount of time as opposed to having to derive new equations by hand and having to modify a whole block diagram.

The generation of MATLAB M files is oriented to produce a generalized model structure that makes it possible to simulate nonlinear systems using MATLAB and also opens the door to the use of other graphical tools such as SIMULINK. This is possible because the code in MATLAB M files representing bond graph models can be used as S files representing blocks with appropriate inputs and outputs in SIMULINK. These correspond to non-linear system blocks or to blocks representing state variable form with multi inputs or multi outputs.

8. REFERENCES


