11

Analysis of Dynamic Systems Using Bond Graph Method Through SIMULINK

José Antonio Calvo, Carolina Álvarez- Caldas and José Luis San Román
Universidad Carlos III de Madrid
Spain

1. Introduction

The dynamic systems analysis, very common in engineering studies, is relatively simple when the steady state behaviour is analyzed, or when the system has few degrees of freedom. However, for complex systems, the problem can be highly complicated and the classic way to establish the behaviour equations becomes inadequate.

In most of the cases, the main concern of engineering students is to establish the mathematical model that represents the dynamic behaviour of the system and how the different parameters influence the system behaviour, because the equations that represent the dynamic of the system are usually partial differential equations, whose solutions require deep mathematical knowledge that in most cases is not available for the students.

The *Bond Graph* technique (Blundell, 1982) is extraordinarily useful to overcome these difficulties. Bond Graph is a simple and effective method to set out the differential equations of any dynamic system independently of the physical field analyzed. *Bond Graph* provides a common model for a wide range of systems ranging from the usual Electric, Mechanics, Hydraulic, Thermals, etc., or combinations of them.

Depcik et al. (2004) compare different software and language options, which are available to build models of dynamic systems. They establish that *MATLAB™* and *Simulink™* might be the best choice if the teacher wishes to collaborate with students because engineering students are typically familiar with *MATLAB*.

Some commercial software allow working directly with *Bond Graph* concepts as CAMP-G, TUTSIM, BONDLAB, which allow drawing the flow lines of the Bond Graph method. However, in order to the students understand the physical and mathematical concepts involved on the dynamic system, the block diagram used in *Simulink* allows a better compression of physical behaviour of the system.

Simulink forms the core environment for Model-Based Design for creating accurate, mathematical models of physical system behaviour. The graphical block-diagram lets the user drag-and-drop predefined modelling elements, connect them together and create models of dynamic systems. These dynamic systems can be continuous-time, multi-rate, discrete-time, or virtually any combination of the three.

In this chapter, we present an application, developed in *Simulink* library, which allows the engineering students to learn easily and quickly about dynamic systems behaviour through *Bond Graph* method.

After a brief introduction to the *Bond Graph* method, it will be explained how *Simulink* can be applied to improve the method, transforming the *Bond Graph* graphical diagram to a *Simulink* block diagram. Next, some mechanical examples of the application will be presented, increasing complexity, to demonstrate the benefits of the method for building complex models from simpler ones. Finally, the results of the dynamic response to different excitations will be analyzed using the various tools provided by *Simulink*.

2. Bond Graph method

This paper does not aim to develop the *Bond Graph* method. There are many literatures about *Bond Graph* method and its applications to analyze dynamic systems, such as Vera et al. (1993), Thoma (1975), Margolis (1985) and Karnopp (1979) in which the method is explained adequately. However, we believe that it is appropriate to make a brief introduction for encourage uninitiated readers to extending the method.
In this paper, the fundamental bases of the *Bond Graph* theory will be presented, in order to understand how to implement a model in Simulink, but is not the focus of the present work to explain the *Bond Graph* method.

2.1 Bond Graph basis

Energy is a basic commodity in a system. It flows in from one or more sources, is temporarily stored in system components or partially dissipated in resistances as heat, and finally arrives at "*sinks*" or "*loads*" where it produces some desired effects. Power is the rate of flow energy and is a scalar with no direction.
Bond Graph represents this power flow between two systems. This flow is symbolized through an arrow (*Bond*) as figure 1 illustrated. Unfortunately, power is not easy to measure directly, and engineers prefer to work with two temporary variables called "*flow*" and "*effort*". Depending on the physical environment, these variables have different value. For example in electrical networks, flow represents the "current" and effort the "voltage", in mechanical linkages, flow represents the "velocity" and effort the "force". The product of both temporary variables is power as equation (1) shows in the case of a mechanical system:

$$Power = f(t) \cdot e(t) = v(t) \cdot F(t) \tag{1}$$

On each *Bond*, one of the variables must be the cause and the other the effect. This can be deduced by the relationship indicated by the arrow direction. Effort and flow causalities always act in opposite directions in a *Bond*.

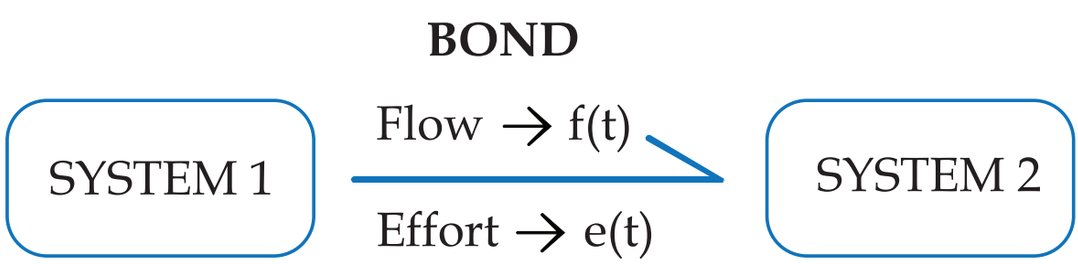


Fig. 1. Power flow in *Bond Graph*

Power bonds may join at one of two kinds of junctions: a "*0*" junction and a "*1*" junction. In a "*0*" junction, the flow and the efforts satisfy the expressions (2):

$$\begin{aligned} &\sum_n Flow_{input\_n} = \sum_m Flow_{output\_m} \\ &Effort_{input\_1} = Effort_{input\_2} = ... = Effort_{input\_n} = Effort_{output\_1} = ... = Effort_{output\_m} \end{aligned} \tag{2}$$

This corresponds to a node in an electrical circuit (where Kirchhoff's current law applies).
In a "*1*" junction, the flow and the efforts satisfy the expressions (3):

$$\begin{aligned} &\sum_n Effort_{input\_n} = \sum_m Effort_{output\_m} \\ &Flow_{input\_1} = Flow_{input\_2} = ... = Flow_{input\_n} = Flow_{output\_1} = ... = Flow_{output\_m} \end{aligned} \tag{3}$$

This corresponds to force balance at a mass in a mechanical system. An example of a "*1*" junction is a resistor in series. In junction, the premise of energy conservation is assumed, no lost is allowed.
There are two additional variables, important in the description of dynamic systems. These variables, often called energy or dynamic variables are: *Momentum* [p(t)] and *displacement* [q(t)].
The variable "*Momentum*" is defined as the time integral of the effort as equation (4) shows.

$$p(t) = \int e(t)\, dt \;\Rightarrow\; \frac{dp}{dt} = e(t) \tag{4}$$

Likewise, the variable "displacement" is defined as the time integral of flow as equation (5) shows.

$$q(t) = \int f(t)\, dt \;\Rightarrow\; \frac{dq}{dt} = f(t) \tag{5}$$

In this way the energy of the system will be:

$$E = \int e(q)\, dq \tag{6}$$

The method makes possible the simulation of multiple physical domains, such as mechanical, electrical, thermal, hydraulic, etc., including combinations of these with each other, which can be treated as a unified set of elements. *Flows [f(t)]* and *efforts [e(t)]* should be identified with a particular variable for each specific physical domain which is working.
Table 1 shows the physical meanings of the variables considered in the general notation and their particular representation for different domains.
Once the system is represented in the form of *Bond-graph*, the state equations that govern its behaviour can be obtained directly as a first order differential equations in terms of generalized variables defined above, using simple and standardized procedures, regardless of the physical domain to which it belongs, even when interrelated across domains.
Bond Graph used a set of elements to model the real system such as:

- **Resistor**: This element represents situations where a lost of energy appears. (Electrical resistor, mechanical damper, Coulomb frictions, etc.). In these sorts of elements there is

a relationship between flow and effort as the equation (7) shows. The value of "R" can be constant or function of any system parameter including time.

$$e(t) = R \cdot f(t) \tag{7}$$

- **Compliance:** This element represents the situations where a store of energy appears (electrical capacitors, mechanical springs, etc.). In these sorts of elements there is a relationship between effort and displacement variable as the equation (8) shows. The value of "K" can be constant or function of any system parameter including time.

$$e(t) = K \cdot q(t) \tag{8}$$

- **Inertia:** This element represents the relationship between the "flow" and Momentum (electrical coil, mass, moment of inertia, etc.) as the equation (9) shows. The value of "I" tends to be constant.

$$p(t) = I \cdot f(t) \tag{9}$$

- **Sources:** This element represents the energy sources. There are two kinds of sources: Flow source and Effort source.
- **Transformer:** A transformer adds no power but transforms it, such as an electrical transformer or a lever. Transformers represent those physical phenomena that are variation of the values of output flow and effort on the values of input flow and effort. If the transformation ratio is given by the "Tr" value, then the relationship between input and output is shown in Equation 10.

$$e_{output}(t) = T_r \cdot e_{input}(t) \qquad f_{output}(t) = \frac{1}{T_r} \cdot f_{input}(t) \tag{10}$$

Physical System	Effort Variable e(t)	Flow Variable f(t)	Momentum Variable p(t)	Displacement Variable q(t)
Mechanical Translation	Force F (N)	Speed V (m/s)	Momentum P (N s)	Displacement X (m)
Mechanical Rotation	Torque M (Nm)	Rotational speed w (Rad/s)	Angular Momentum H (N m s)	Angle q (Rad)
Electrical Systems	Voltage V (V)	Intensity I (A)	Current Flow F (V s)	Electrical Charge q (C)
Hydraulic Systems	Pressure P (N/m2)	Flow Q (m3/s)	Momentum R (N s/m2)	Volume V (m3)

Table 1. Generalized variables for different physical domains and units

Figure 2 shows the *Bond Graph* representation of the different basic elements available to model a system.

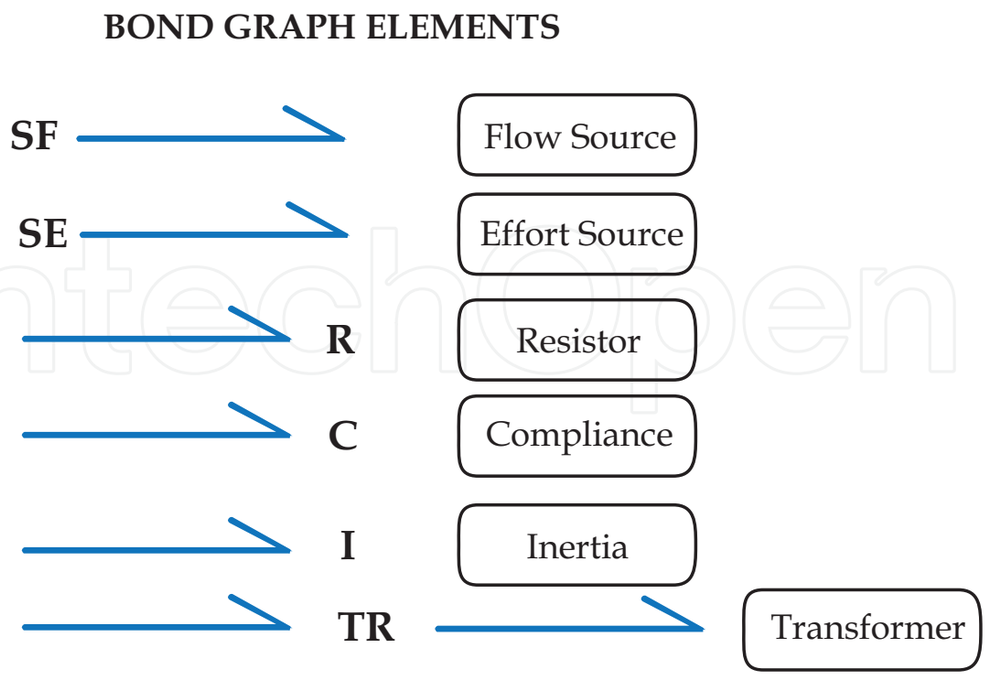


Fig. 2. *Bond Graph* basic elements

There are more elements in the *Bond Graph* method, but those showed above are enough for the following examples.

2.2 Mechanical example

In order to understand the *Bond Graph* method, a simple mechanical model will be used. Figure 3 illustrates a single degree of freedom (DOF) system composed by a rigid body that can only move up and down. This model is represented by a mass "m_1", a spring (K_1) and a damper (R_1). The source of energy is a known velocity of the ground $v_0(t)$. The output variable is the mass velocity $v_1(t)$.

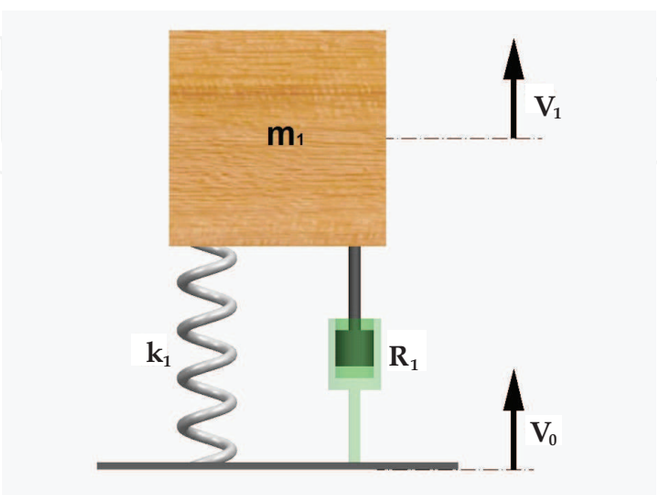


Fig. 3. One degree of Freedom System

Usually, the starting point to represent a real system as a *Bond Graph* diagram are the sources, in this case a flow source (ground velocity). The next step is to think about the power flows through the system components, power flows produced by the effort and flow variables. As figure 3 illustrates the spring and the damper are moved at a velocity that must be the difference between the mass velocity $v_1(t)$ and ground velocity $v_0(t)$. This means that a "*0*" junction must be used as figure 4 illustrates.

The spring is represented as a compliance bond and the damper as a resistance bond. Both components move at the same velocity, but each one needs different forces to move at this velocity. This means that a "*1*" junction must be used as figure 4 illustrates. Finally, the mass is represented as an inertia bond that moved at $v_1(t)$ velocity. Due to the fact that the system is under gravity of earth, an effort source must be used.

Bond Graph method calculates the flow and effort on each *Bond*, and uses as system variables the displacement associated to compliance (K_1) bond *[x(t)]* and momentum associated to inertia (m_1) b*ond [P(t)].*

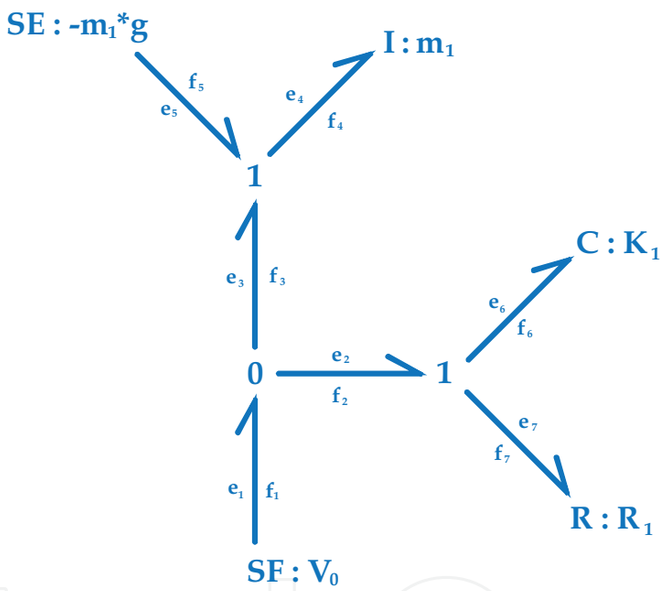


Fig. 4. *Bond Graph* of the one DOF system

According to the *Bond Graph* showed in figure 4, the flows in each bond are presented in the equations shown in (11):

$$\begin{aligned} f_1 &= V_0(t) \\ f_2 &= f_1 - f_3 = V_0(t) - \frac{P}{m_1} \\ f_3 &= f_4 = f_5 = \frac{P}{m_1} \\ f_6 &= f_7 = f_2 = V_0(t) - \frac{P}{m_1} \end{aligned} \tag{11}$$

The efforts in each bond are presented in the equations shown in (12).

$$
\begin{aligned}
&e_1 = e_2 = e_3 \\
&e_2 = e_6 + e_7 = K_1 \cdot X + R_1 \cdot \left(V_0(t) - \frac{P}{m_1} \right) \\
&e_7 = K_1 \cdot X \\
&e_6 = R_1 \cdot \left(V_0(t) - \frac{P}{m_1} \right) \\
&e_5 = -m_2 \cdot g \\
&e_4 = e_5 + e_3 = K_1 \cdot X + R_1 \cdot \left(V_0(t) - \frac{P}{m_1} \right) - m_1 \cdot g
\end{aligned}
\tag{12}
$$

That system dynamic behaviour is represented by the following two equations shown in 13 and 14:

$$\frac{dX(t)}{dt} = f_7 = V_0(t) - \frac{P(t)}{m_1} \tag{13}$$

$$\frac{dP(t)}{dt} = e_4 = K_1 \cdot X(t) + \left(V_0(t) - \frac{P(t)}{m_1} \right) \cdot R_1 - m_1 \cdot g \tag{14}$$

These equations are achieved taking into account that the variation of the momentum is equal to the effort on the inertia bond, and the variation of the displacement is the flow in the compliance bond. These are two first order differential equations coupled as a function of the momentum [P(t)] associated to the inertia and displacement [X(t)] associated to the spring. These systems can be solved without major problems by using differential calculus.

3. Introduction to Simulink

Simulink is an environment for multidomain simulation and Model-Based Design for dynamic and embedded systems. It provides an interactive graphical environment and a customizable set of block libraries that allow to design, simulate, implement, and test a variety of time-varying systems, including communications, controls, signal processing, video processing, and image processing.

Simulink is integrated with *MATLAB*, providing immediate access to an extensive range of tools that make possible to develop algorithms, analyze and visualize simulations, create batch processing scripts, customize the modelling environment and define signal, parameters and test data.

With *Simulink*, a detailed block diagram of a system can be quickly created, modelled and maintained using a comprehensive set of predefined blocks. *Simulink* provides tools for hierarchical modelling, data management, and subsystem customization, making it easy to create concise, accurate representations, regardless of your system's complexity. *Simulink* software includes an extensive library of functions commonly used in modelling as:

- Continuous and discrete dynamics blocks, such as Integration and Unit Delay
- Algorithmic blocks, such as Sum, Product, and Lookup Table
- Structural blocks, such as Mux, Switch, and Bus Selector

Working directly in Simulink, these built-in blocks can be modified and new ones can be created and placed into customized libraries.
After building a model in *Simulink*, its dynamic behaviour can be simulated, and live results can be viewed. *Simulink* software provides several features and tools to ensure the speed and accuracy of the simulation, including fixed-step and variable-step solvers, a graphical debugger, and a model profiler.
All these features of the program allow generating a block model suitable to represent the *Bond Graph* model. The *Simulink* block model will solve the equations and allows the user to analyze the dynamic behaviour of the system.

4. Simulink application

After previous discussions about *Bond Graph* method and Simulink, this section explains how to use the benefits of *Simulink* to set up and solve the equations that manage system behaviour.
The procedure consist in convert the real model into a *Bond Graph* model and then translate it to the *Simulink* block diagrams, as can be seen in Figure 5.

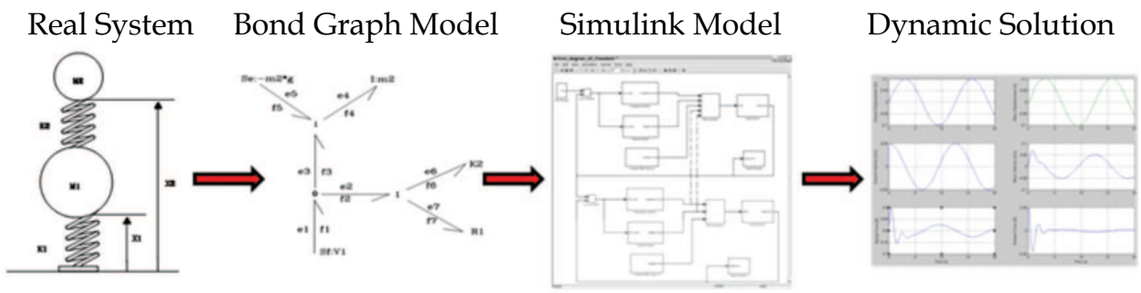


Fig. 5. Procedure to Analyze the Dynamic Behaviour of a System

Simulink provides a block-diagram that allows users to represent the equations involved on the *Bond Graph* model. These blocks of *Simulink* will be used to represent the action that occurs in each element of the *Bond Graph* model.
The different *Simulink* blocks that will be used to build the *Bond Graph* model are being described following.
It is assumed that readers know the basic operation of *MATLAB* and *Simulink*, so no details will be given regarding how to open the program, select Simulink blocks and build models from these *Simulink* blocks.
Figure 6 shows the main window of *Simulink* to build a new model, and the available blocks that have been selected. As it can be seen, very few blocks are needed to generate models.
From the working window of *MATLAB, Simulink* is accessed by clicking the icon at the top of the screen. Then, the user has to select open a new model and a new workspace appears.
From the *Simulink* library, the necessary blocks are selected and dragged to the working window. The blocks used for simple models are as follows:

- **Sources library:** *Time, Constant and Signal Generator.* These blocks provide an interface that allows the user to input energy into the system through a known flow or effort. *Simulink* provides different kinds of sources such as constant value, sinusoidal source, chirp signal, ramp, etc.

- **Continuous library:** *Derivate and Integrator*. These blocks provide an interface that allows to derive or integrate time-dependent variables of the dynamic system.
- **Math Operation library**: *Add, Divide, Product*. These blocks provide an interface that allows to add, subtract, multiply, divide, etc., time-dependent variables of the dynamic system.
- **Sinks library**: *XY Graph and To Workspace*. These blocks provide an interface that allows to display and store the variables of the dynamic system in order to analyze them.

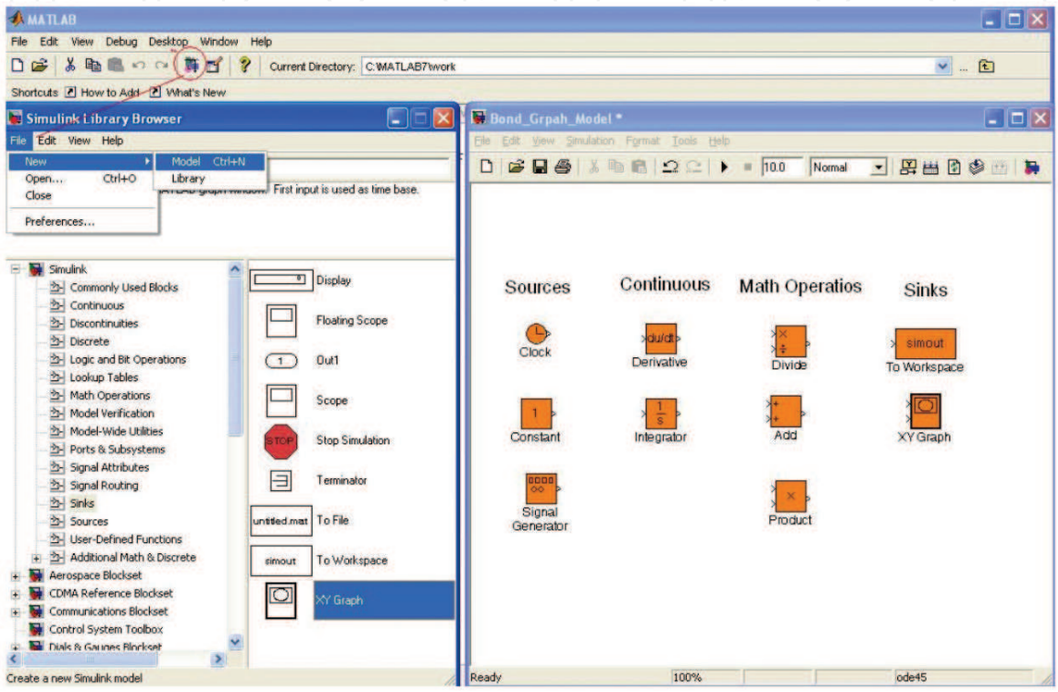


Fig. 6. Simulink working model window

Although there are many more blocks that the user can select and use to generate models, only those previously described are used in this work.

Taking into account the effect of each *Bond Graph* element on the flow and effort variables, the idea is to generate *Simulink* elements that perform the same actions.

Bond Graph and *Simulink* allows to work with discrete time dynamic systems, for instance, it is possible to simulate a variable damping shock absorber with two damping states (hard and soft). These two states could change as a function of other variable of the system.

On the other hand, as the equations and variables used for all physical systems are the same, it is possible to combine, in one model, mechanical and electrical systems or mechanical and hydraulic, etc. No changes are necessary in Simulink, just selecting the adequate block.

The first step is to select the necessary blocks to simulate the *Bond Graph* element behaviour. Then, it is possible to generate a new Simulink block through the "*sub-system create*" feature, that groups blocks to generate a user-defined one. Figure 7 illustrates how to create a subsystem: select the group of blocks to generate the subsystem and with the right button of the mouse, select the option "*create subsystem*". Then a new "*Subsystem*" block appears.

In order to facilitate the user to enter parameter values, another *Simulink* feature called "*mask subsystem*" can be used.

The path to enter this feature is to click the right button of the mouse on the *Subsystem block* and select the option "*edit mask*". The window shown in Figure 8 will appear and the user has to select the tab "*parameters*". By clicking on the first icon on the left, a new line is added. This line allows entering the name of the variable and its abbreviation. It must be matched with what it was entered in the original block. It is necessary to define as many variables as defined in the *Subsystem* and variable names must be kept. Variable name must be introduced in the label called "Variable", and the label "Prompt" allows to introduce a variable description.

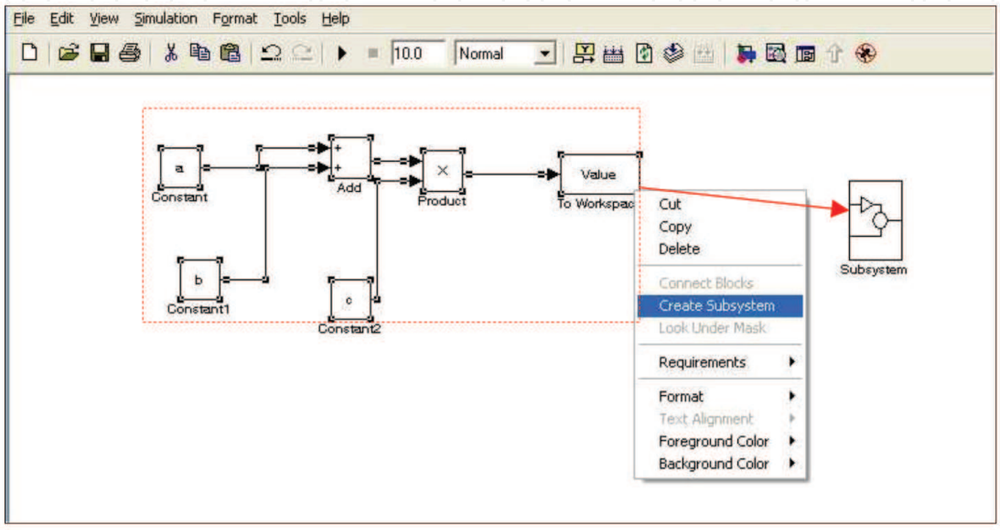


Fig. 7. Subsystem generation

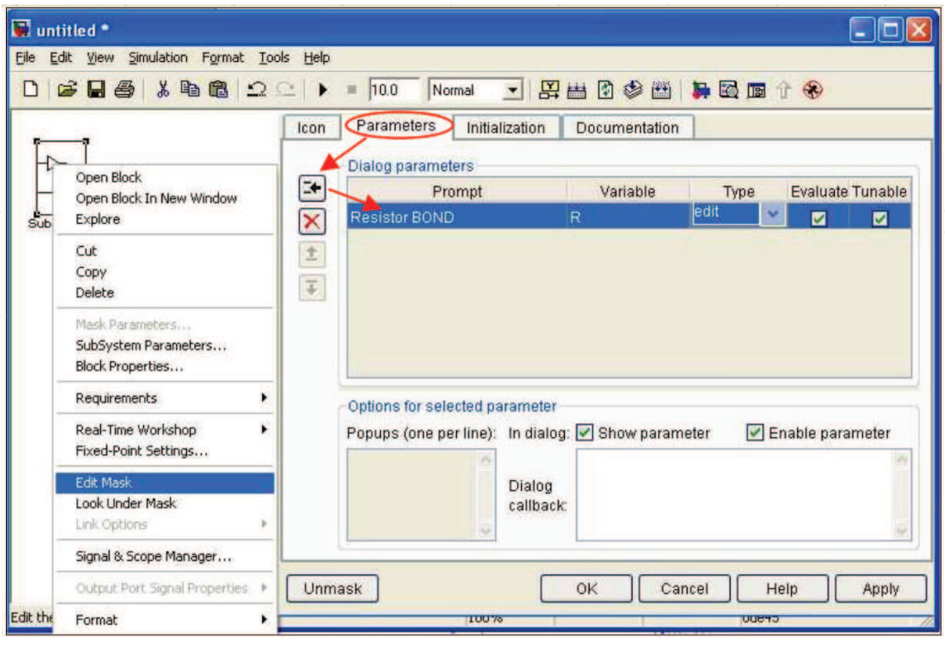


Fig. 8. Mask editor

Thus, when the user clicks on a *Subsystem block*, a new window appears to enter the parameter value as shown in Figure 9.

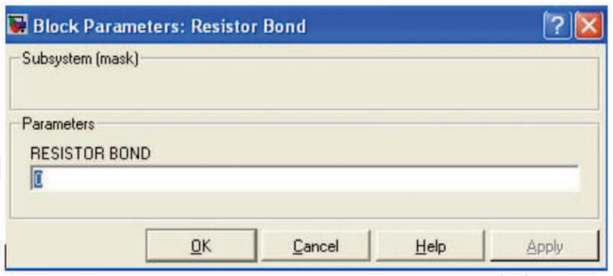


Fig. 9. Editor window for parameters

With exposed blocks, it is possible to generate the different elements to create a *Bond Graph* model following the next rules:

Source bond

This element, as has been already indicated, provides energy to the system, as a known flow or a known effort. So, the *Simulink* source blocks can be used directly.

Resistor bond

It provides an interface that allows the user to simulate a component that dissipates energy from the system. The input block is a known flow that is multiplied by the resistance value in order to obtain the output, which is a known effort as Figure 10 illustrates. The R value could be constant or variable. In each case, the necessary block is different (For instance a "*Lookup Table*").

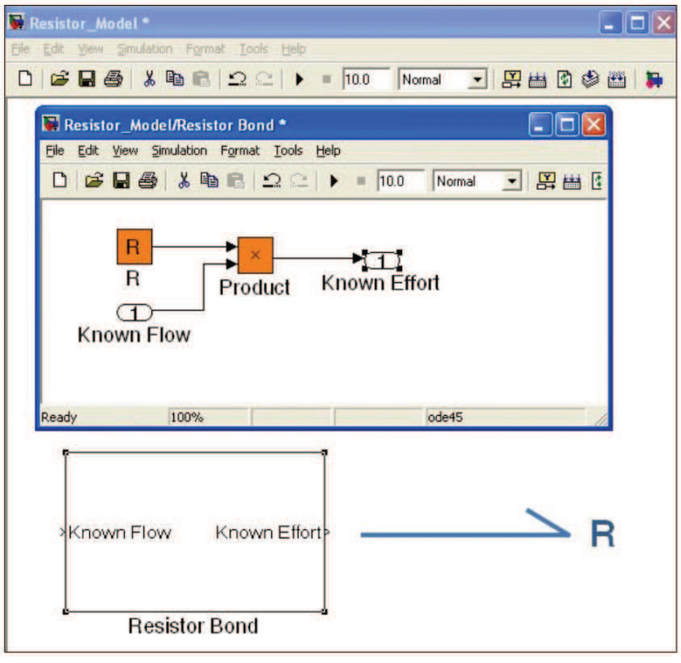


Fig. 10. Resistor bond

Compliance bond

It provides an interface that allows the user to simulate a component that stores energy from the system. The input block is a known flow that is integrated in order to obtain the displacement and multiplied by the compliance value to return a known effort as Figure 11 illustrates. To establish the initial position of the inertia at the beginning of the simulation, it is possible to preload the compliance by adding the initial displacement.

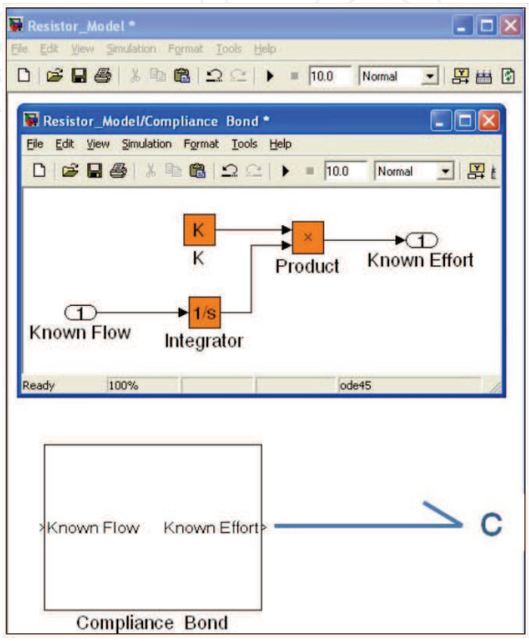


Fig. 11. Compliance bond

Inertia bond

It provides an interface that allows the user to simulate the system inertias. The input block is a known effort that is divided by the inertia value to obtain the flow derivate as equation (15) shows.

$$e(t) = I \cdot \frac{df(t)}{dt} \Rightarrow f(t) = \frac{1}{I} \cdot \int e(t)dt \tag{15}$$

Then, the value is integrated to return to a known flow as illustrated in Figure 12.

Junction "1" and "0"

They provide an interface that allows the user to simulate the junctions to sum flows or efforts. Both blocks are simple *Simulink's "add"* blocks. It is possible to add more input connections.

Transformer bond

It provides an interface that allows the user to simulate elements that modify the value of the flow and the effort. The input blocks are a known flow and effort, multiplied by the transformer ratio. Its inverse is calculated to obtain the flow and effort at the output. Figure 13 illustrates the block diagram.

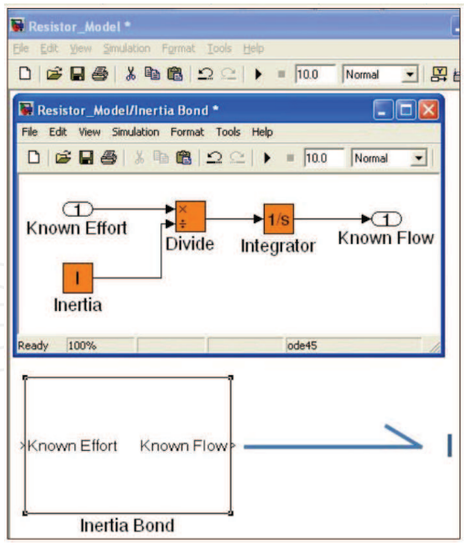


Fig. 12. Inertia bond

Display block

It provides a block that allows the user to store the system's variables in order to analyze and visualize results once the simulations are completed.

Finally, the working window with the blocks needed to create a model is presented in Figure 14. Three new blocks has been added, one of them (*XY Graph*) to display the variables of the model during the simulation at real time, the second one (*Simout*) to store the results in computer memory to be viewed later and the third one is a clock to manage the simulation time.

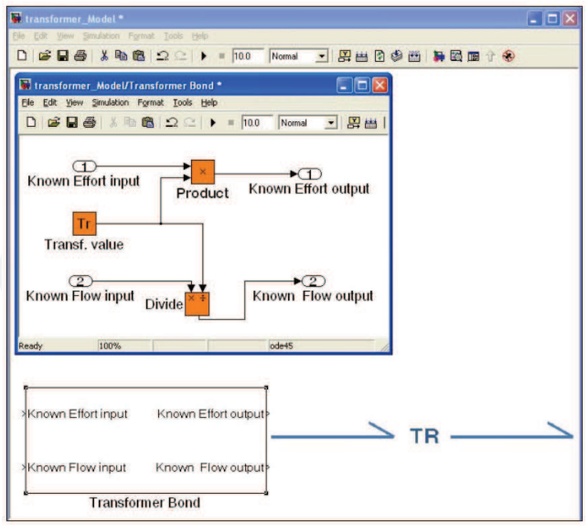


Fig. 13. Transformer bond

Finally, through *Simulink* interface, it is possible to select the method to integrate the equations, the simulation time and the integration step in order to adapt them to the user needs.

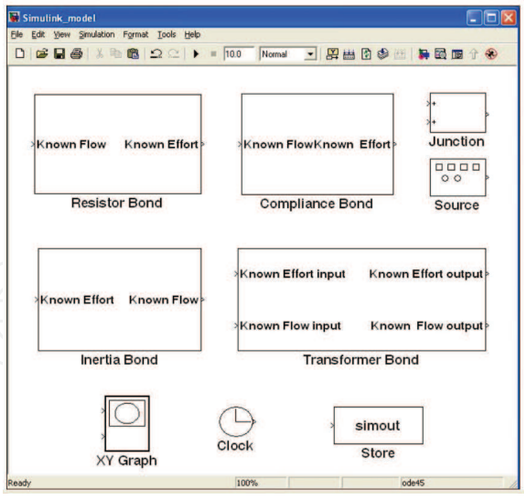


Fig. 14. Simulink Blocks built from Bond Graph Elements

5. Application example

This section illustrates how *Bond Graph* and *Simulink* were used by mechanical engineering students to analyze, in an easy way, the dynamic behaviour of some typical systems.
The analysis begins with the example of a one degree of freedom system, like the one used to introduce the *Bond Graph* method in the previous section (Figure 3). Table 2 shows the model parameters used as inputs.
Units are expressed in international unit system. It is assumed that the positive displacement of the mass is upward. Because of that, the weight force is negative.

Parameter	Value	Unit
Mass (m_1)	1	kg
Spring ratio (K_1)	9.8	N/m
Spring initial position (X_0)	1	m
Damper ratio (R_1)	1.88	N s/m
Weight (Se)	-9.8	N
Power Source System (Sf)	Variable	m/s

Table 2. Parameters for a One Degree of Freedom System

Figure 15 illustrates the *Simulink* block diagram of the one degree of freedom system based on *Bond Graph* method developed in Figure 4.
The *Flow* source (ground velocity) is connected to a sum block (*1 junction*) that calculates the difference of velocities between the ground (V_0) and the mass (V_1). This velocity is the input (known flow) for the spring (*Compliance Bond*) and the damper (*Resistor Bond*), that move at the same speed. They return to the system the spring force (F_k) and the damper force (F_r) that are connected to a sum block (*0 junction*) to calculate, by adding gravity force (*Effort Source*), the equilibrium force. This force will be the input to the *Inertia block* in order to calculate the velocity of the mass (V_1) that is used as an iterative loop to calculate the difference in velocity between the ground and the mass.

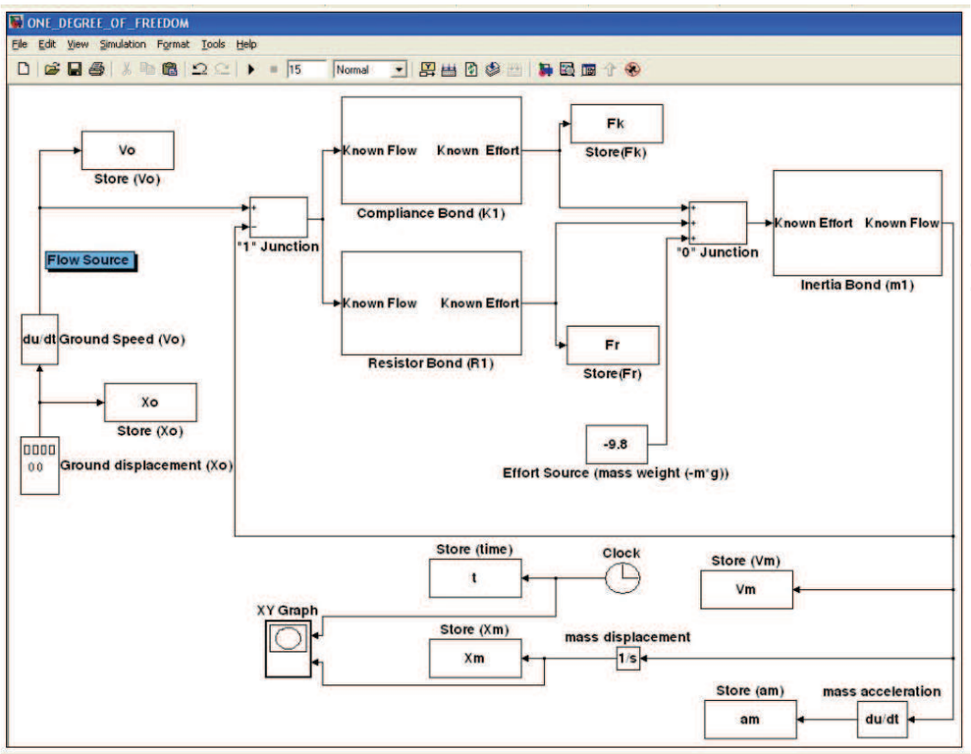


Fig. 15. One Degree of Freedom Model

The analysis parameters are the following:

- F_k Spring Force.
- F_r Damping Force
- X_m Mass Displacement
- V_m Mass Speed
- a_m Mass Acceleration
- X_0 Ground Displacement
- V_0 Ground Speed
- t Time

All of them have been stored in variables with their own names in order to analyze them and compare results by plotting graphs.

Due to the fact that the input to the system must be a flow (ground speed) it is necessary to derivate the ground displacement in order to obtain a flow source.

To decide the kind of solver that will be used to integrate the equations, it is necessary to select the *simulation* menu and then select the option *configuration parameters*. A new window will open as figure 16 shows. Simulation time, solver options and the sample time can be introduced in this window. For instance, the *Runge-kutta* integration method has been selected in this case, with a fixed step of 0.01 second. This method has been selected because it works with first-order differential equations and the convergence of this method is good without excessive computer time consumption, but any other method available in Simulink can be chosen since it will not influence the quality of the result. A step of 0.01 s allows to analyze the result with enough resolution to apply, for instance, a Fast Fourier Transform (FFT) without loss of information; but the user can try to adjust the best in each case.

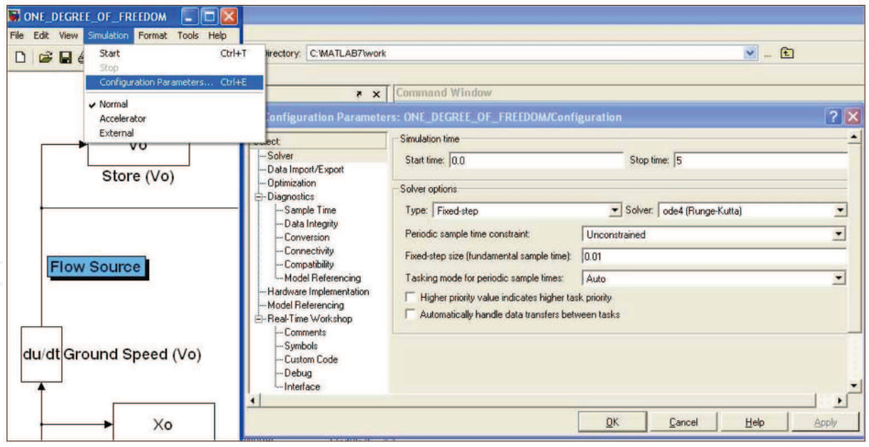


Fig. 16. Solver selection

Four different situations of the example model will be analyzed to show the advantages of using *Simulink* to generate, solve and analyze the results.

5.1 Free vibration

Firstly, the ground is stopped (v_o= 0 m/s) and the spring is not preloaded (Xo = 0 m), this means that the mass will fall on top of the spring and therefore the system moves (this is known as free vibration). Under the damper effect, the system will stop after a few cycles due to the lost of energy in each cycle.

The powerful tool of *MATLAB* to plot the results, the *MATLAB* Graphics Editor, will be used. Every variable has been stored in *MATLAB* workspace and is available to be represented. To access the graphics editor, the workspace flange must be selected in the *MATLAB* window, as figure 17 shows. Every available variable is showed in this window. The desired variable is selected with the button of the mouse. By clicking in the plot icon, a new window with the variable plotted will open. Then, it is possible to modify the chart by adding new variables, changing the scale or including legends.

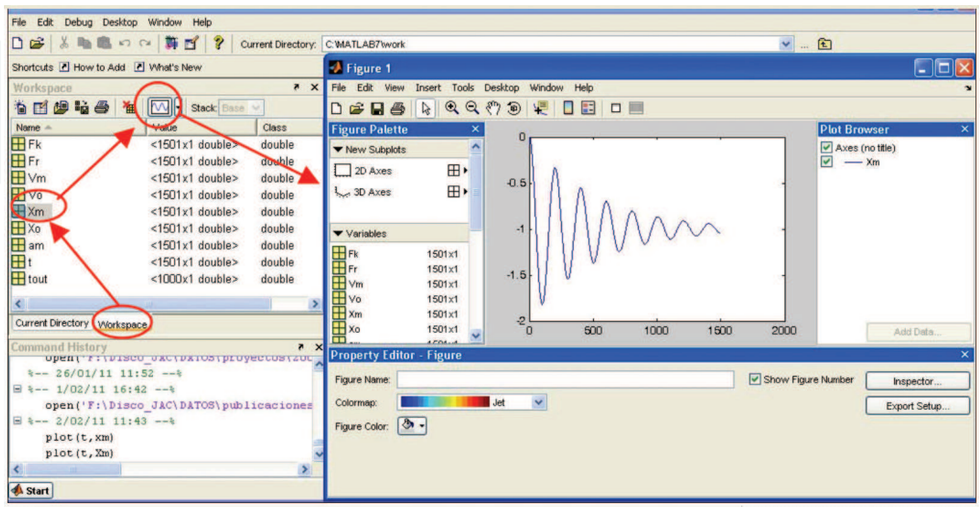


Fig. 17. *Matlab* graphics editor

Figure 18 illustrates the system behaviour. Different values have been analyzed, such as: ground displacement and velocity, mass displacement and acceleration and the force in the spring and the damper.

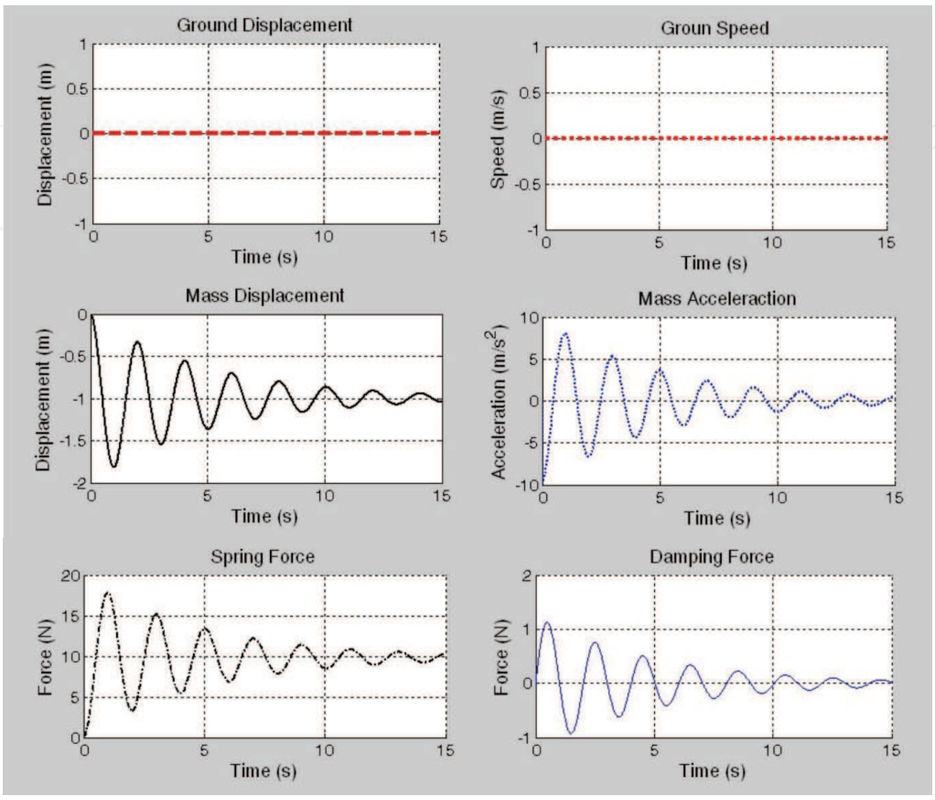


Fig. 18. Results for the One Degree of Freedom Model with no Ground Displacement

By looking the different graphs of the variables that have been plotted, it is possible to analyze what is happening to the model when the mass can move freely.
Initially, the mass falls and compresses the spring. Then, when the inertia and gravity forces achieve a value equal to the spring force, the mass returns to the steady state position. This movement is repeated but, as consequence of the damper, the system loses energy and finally stops after few cycles. When the system stops, the spring is loaded with a force equal to the mass weight (9.8 N) so the mass stabilizes at minus one meter from the initial position.

5.2 Forced vibration below natural frequency

In the second simulation, the system starts with the spring preloaded (Xo = 1m), and the ground has a sinusoidal movement with amplitude of 0.1 meter and a frequency of 0.1 Hz (This is known as forced vibrations) The natural frequency of the system, according to the equation 16, is under the frequency of the excitation

$$F(Hz) = \frac{1}{2\pi}\sqrt{\frac{K_1}{m_1}} = 0.498Hz \tag{16}$$

To change the model parameters, the user has to click on the block and put the new values. Taking into account that the system excitation is under the natural frequency, the mass movement will be similar to the ground movement as figure 19 illustrates.
As figure 19 shows, the system movement has a transitional state before stabilizing and oscillating around the initial position. Since the difference between the mass velocity and the ground velocity is very small, the energy dissipated by the damper is small too.

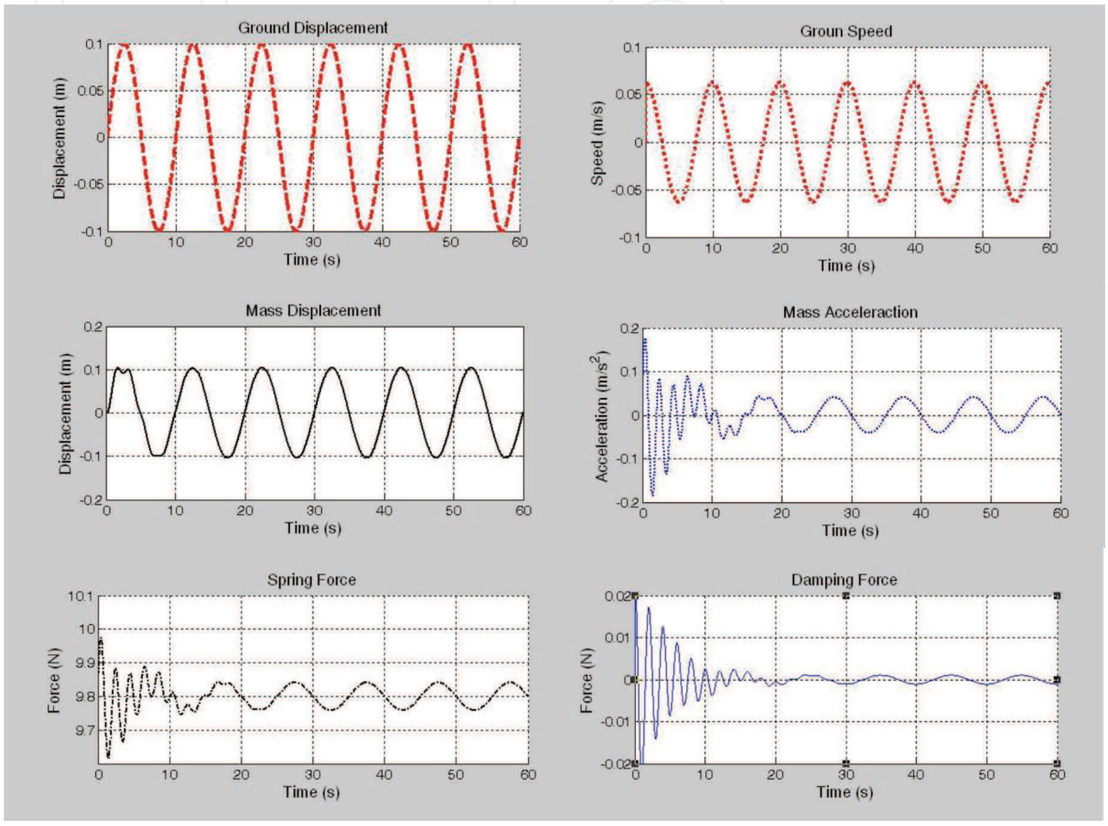


Fig. 19. Results for the One Degree of Freedom Model with Ground Displacement below the Natural Frequency

5.3 Forced vibration at natural frequency

In the third simulation, the sinusoidal movement of the ground is the natural frequency of the system. In this case, the movement of the mass is seven times higher than the ground movement due to the resonance phenomenon, as figure 20 illustrates. Again, after a little transitional period, the system achieves the steady state. Now, the energy dissipated on the damper is high as well as the force at the spring

5.4 Forced vibration up to natural frequency

In this simulation, the ground moves at a frequency (2 Hz) that is higher than the natural frequency of the system. Figure 21 illustrates the variable analysis.
In this case, the mass movement is lower than the ground movement, but as both of them are out of phase, the damper has to dissipate more energy than in the case analyzed in 5.2.

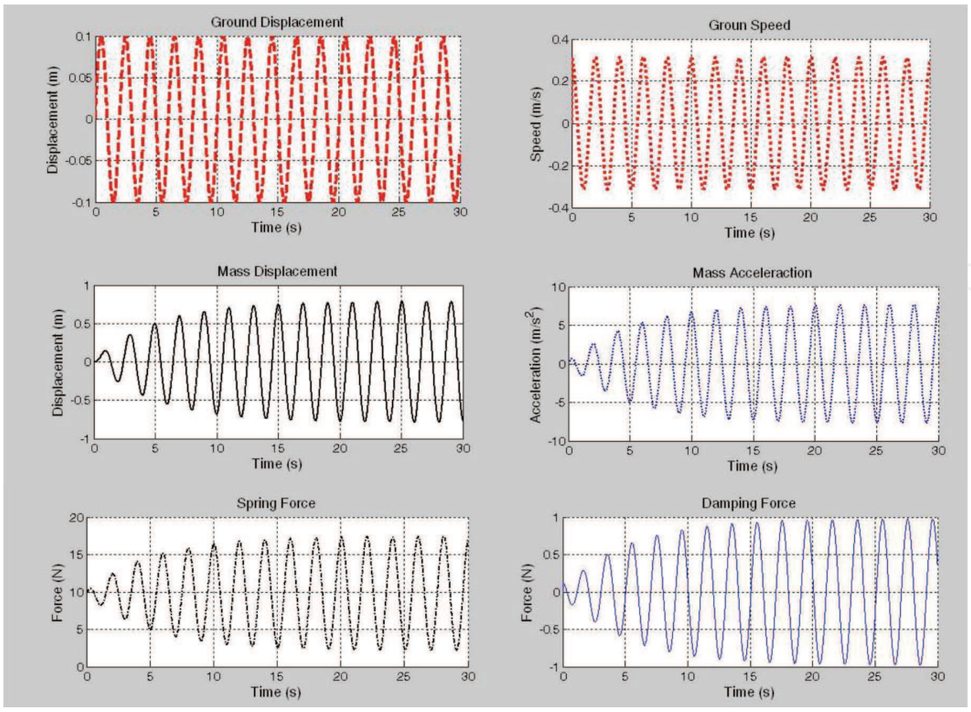


Fig. 20. Results for the One Degree of Freedom Model with Ground Displacement at Natural Frequency

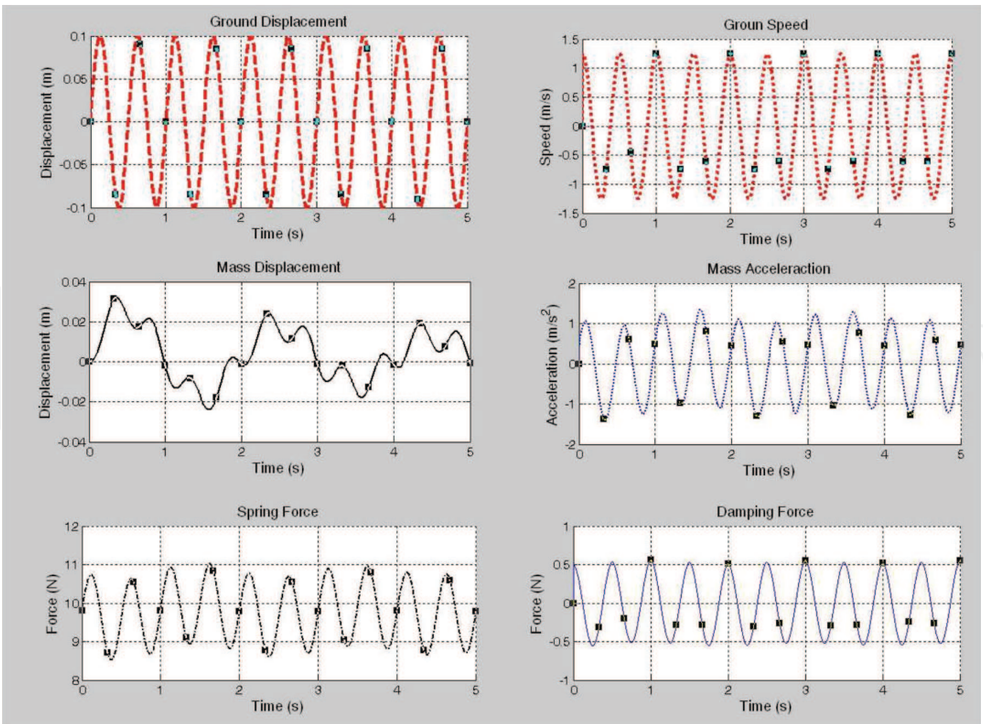


Fig. 21. Results for the One Degree of Freedom Model with Ground Displacement greater that the Natural Frequency

6. Two Degrees of Freedom Model

Finally, a more complex model will be simulated to demonstrate one of the advantages of the *Bond Graph* method and *Simulink* application. From the one degree of freedom model, the complexity of the system will be increased by adding an additional degree of freedom.

Figure 22 illustrates the physical model for a two degrees of freedom system created by adding two single degree of freedom systems.

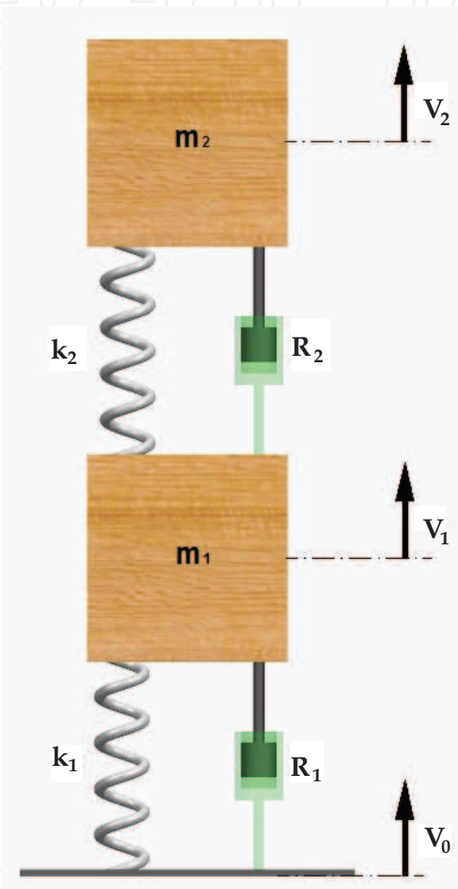


Fig. 22. Two Degrees of Freedom System

Figure 23 illustrates the *Bond Graph* model for the two degrees of freedom system. The system has been developed from the original one of one degree of freedom. The new part has been drawn in red:

As illustrated in figure 24, *Simulink* model it is relatively easy to built since the user only has to copy and paste the original model to obtain a system with two degrees of freedom.

The new part of the model is drawn in red. Only few lines, drawn in blue, are necessary to connect both parts of the model. The new blue lines transmit the effort from the spring (K_2) and the damper (R_2) to the "*zero junction*" that feed the inertia 1(m_1). The inertia 1 block returns a known flow to a "*one junction*" in order to calculate the velocity difference between the two masses that feed the spring K_2 and damper R_2.

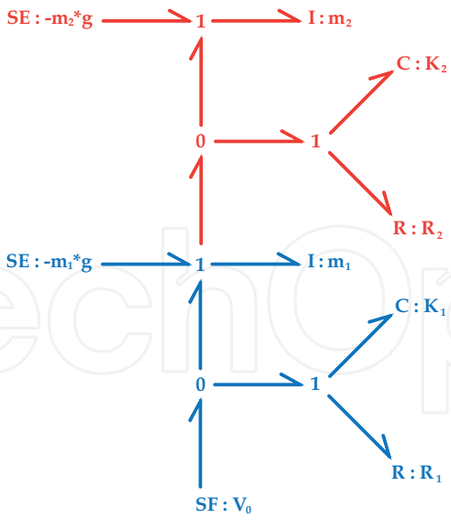


Fig. 23. Bond Graph chart of the Two Degree of Freedom System

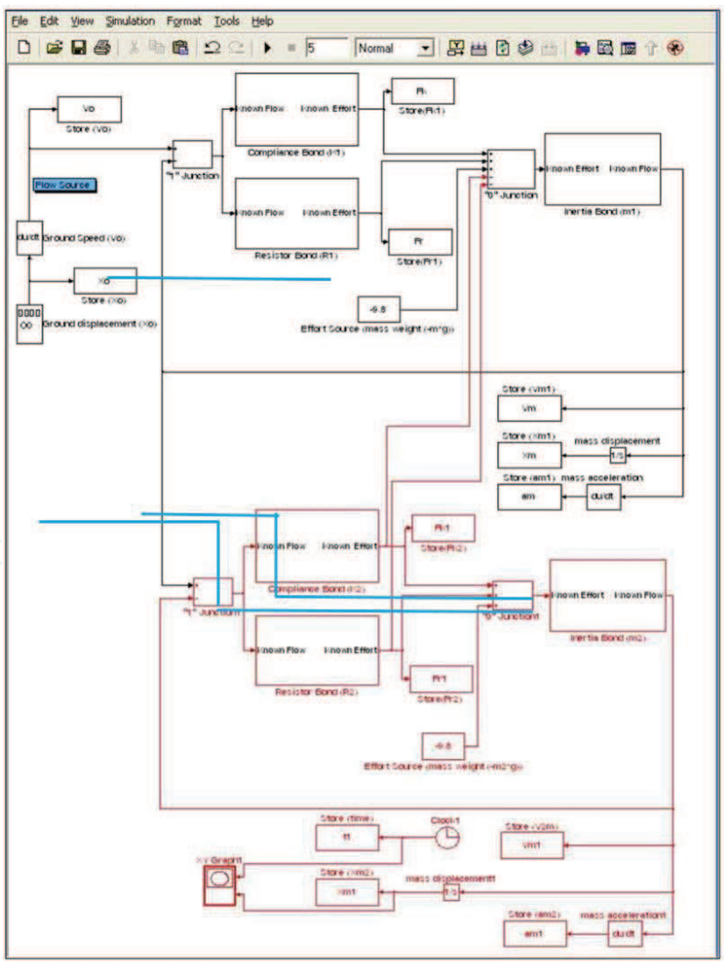


Fig. 24. Simulink blocks of a Two Degrees of Freedom System

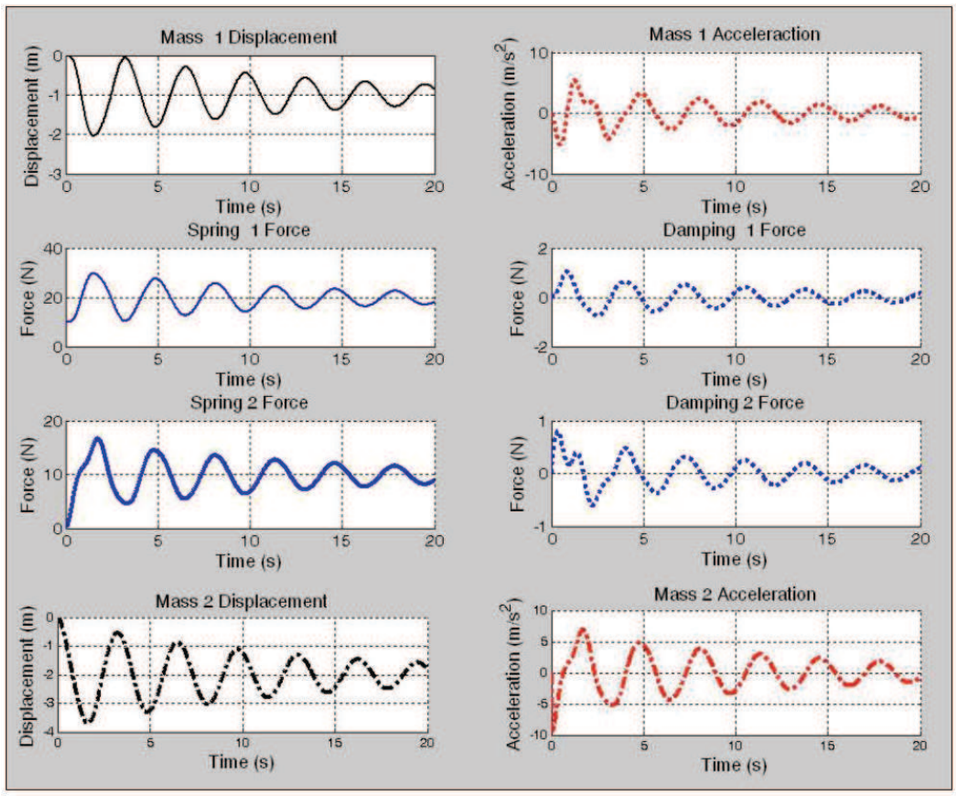


Fig. 25. Two Degrees of Freedom Results

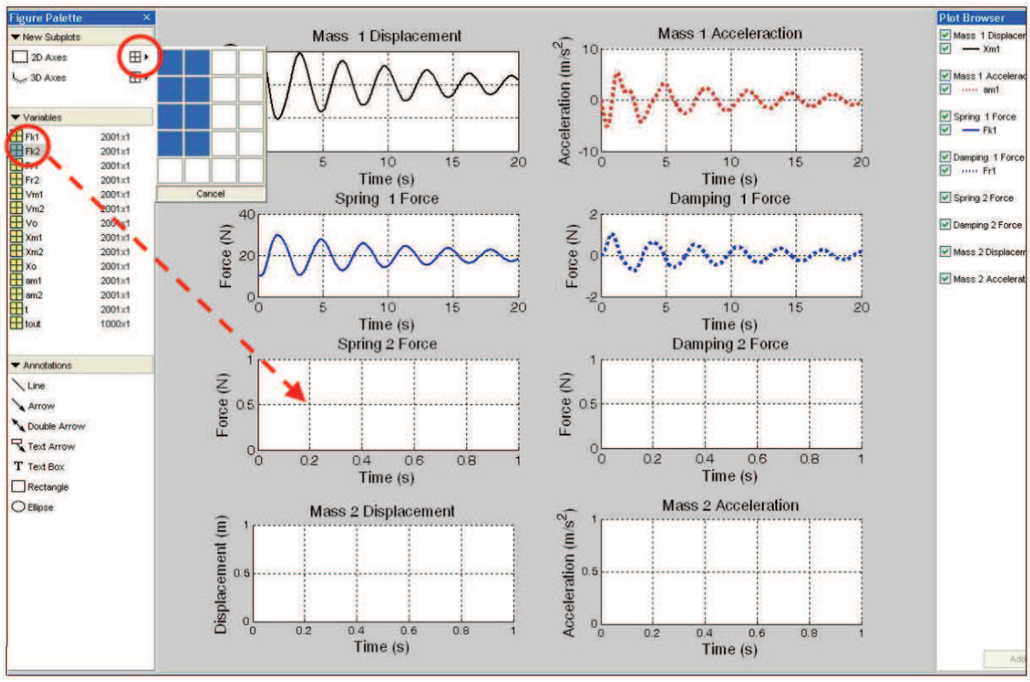


Fig. 26. How to add more plots

It will be necessary to rename the variables of the second part of the model. This requires accessing the "*mask editor*" by clicking with the right button of the mouse in each block, and changing the variable name (m_2, K_2, R_2 …).
The model will be analyzed assuming that initially the ground is stopped, the spring 1 is preloaded, the spring 2 unloaded and mass m_2 falls over it. Figure 25 illustrates the results, analysing the mass displacement and the forces on the springs and the dampers.
In order to obtain the new graphics, the user has to open the plot window and add as many graphics as needed by clicking on the icon "*created titled subplots*" as figure 26 shows.
After that, the desired variable must be clicked and dragged to the blank plot window in which it is going to be represented.
Likewise, by changing model parameters, different inputs can be simulated and the results can be comfortably analyzed.
On the other hand, more complex models can be generated by adding more degrees of freedom following the procedure shown in this example.

6. Conclusion

This work presents a method to apply the *Bond Graph* technique to implement and solve the dynamic equations of a dynamic system by taking advantage of *MATLAB* and S*imulink*.
This method allows engineering students to quickly and easily gain experience and knowledge in systems dynamics and to learn which are the forces, accelerations, velocities and displacements of each component and each degree of freedom of the system.
It is very easy for the students to test the changes in the system and analyze how the results change.
The proposed method is designed with a user-friendly Windows interface in *MATLAB's Simulink*. The most important benefits of using the proposed method are the following:

- **MATLAB** toolboxes and functions can be used, allowing the program to simulate complex systems.
- It is easy to work with differential equations, matrixes and vectors.
- Students do not have to determine the dynamic equations, since the *Bond Graph* method allows to transfer from the graphics model to the block model the equations involved in the dynamic problem.
- *MATLAB* tools are easy to use when analyzing and comparing the model's results by numerical or graphics outputs.

It is noticeable the facility to generate complex models from simples ones and the facility to change the model parameters in order to obtain different results.
The user does not need to have a deep knowledge of differential equations to develop the expressions that represent the behaviour of the system and to solve them.
On the other hand, the *MATLAB* tools allow to generate models with complex behaviour, for instance, dampers with non-linear behaviour.

7. References

Blundell, A. (1982). *Bond Graph for modelling Engineering Systems*, Ellis Horword Publishers., UK

BONDLAB.PH (2007). Department Silicon Facility (DSF). http://ssd-rd.web.cern.ch/ssd-rd/bondlab/default.htm, (September 2007)

CAMP-G. (2007) *Computer Aided Modelling, Design and Simulation*, http://www.bondgraph.com, (September 2007)

Depcik, C. and Assanis, D.N., (2004). *Graphical user interfaces in an engineering educational environment*, Computer Applications in Engineering Education, Vol. 13, No. 1, pp. 48–59.

Karnopp, D., (1979). *On the Order of a Physical Model*, Trans. ASME Journal of Syst. Dyn. Meas. & Control. Vol 101, nº 3, pp. 102 – 118, ISSN: 0022-0434, New York, USA

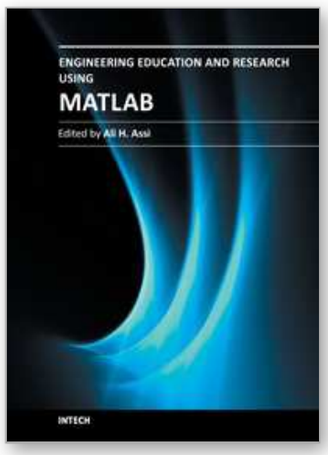
Margolis, D. (1985). *Introduction into Bond Graph,* 3rd Seminar on Advanced Vehicle Dynamics. Amalfim Italy

The MathWorks Inc., (2009). *Creating Simulink Models*, http://www.mathworks.co.kr/matlabcentral/newsreader

Thoma, J.U., (1985). *Introduction to Bond Graph and their applications,* Pergamon Oxford. ISBN: 0080239366, New York, USA

TUTSIM (2007). *Technical University of Twente SIMulation,* University of Twente. http://www.20sim.com/tutsim.html, (September 2007)

Vera, C., Aparicio, F., Felez, J. (1993). *Simulación de sistemas dinámicos mediante la técnica del Bond Graph,* Sección de Publicaciones de la E.T.S.I.I. de Madrid, Universidad Politécnica de Madrid, ISBN: 84-7884-082-1, Spain



Engineering Education and Research Using MATLAB

Edited by Dr. Ali Assi

ISBN 978-953-307-656-0

Hard cover, 480 pages

Publisher InTech

Published online 10, October, 2011

Published in print edition October, 2011

MATLAB is a software package used primarily in the field of engineering for signal processing, numerical data analysis, modeling, programming, simulation, and computer graphic visualization. In the last few years, it has become widely accepted as an efficient tool, and, therefore, its use has significantly increased in scientific communities and academic institutions. This book consists of 20 chapters presenting research works using MATLAB tools. Chapters include techniques for programming and developing Graphical User Interfaces (GUIs), dynamic systems, electric machines, signal and image processing, power electronics, mixed signal circuits, genetic programming, digital watermarking, control systems, time-series regression modeling, and artificial neural networks.

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen