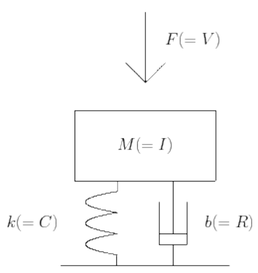
Engineer In a Box

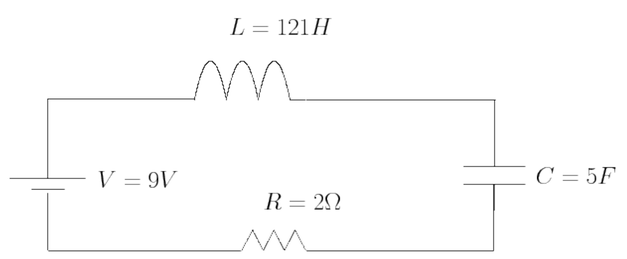
Dylan Visher, Galina Meyer, John Christian
Mr. Anderson - Project Advisor

Modeling systems is an important step of the engineering process. Mathematical and graphical models allow the behavior of a system to be predicted before construction, enabling modifications to be made earlier and reducing costs. However, there is no simple way to model a system involving multiple physical domains, such as electrical, mechanical, and rotational. To create a mathematical model of such a system, an engineer must do a substantial amount of physics to create differential equations, then solve the resulting complex system of equations. This program takes advantage of bond graphs, a uniform, abstract representation of a system, to simplify the modeling process. Bond graphs allow electrical, mechanical, and rotational systems to be modeled through the same abstract process. This process occurs in three phases: data entry in bond graph form, development of a system of equations, and equation solving to produce a graphical output of the system's behavior. This poster presents an implementation of this system applied to a series RLC circuit as well as its mechanical equivilant.

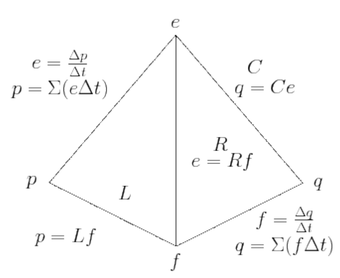
This is a graphical representation of a mechanical system. This specific system has a mass, spring, and damper, and is pushed down by a force. The spring and mass cause an oscillatory, simple harmonic motion. The damper resists simple harmonic motion, eventually causing the mass to come to rest.



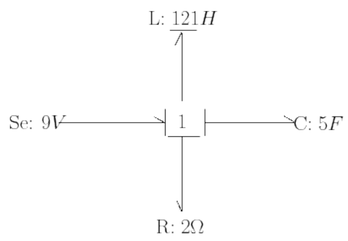
This is a traditional graphical representation of an electrical system. An RLC circuit is composed of a resistor, inductor, and capacitor wired in series to a voltage source. The combination of an inductor and capacitor cause the system to oscillate. The correct behavior of this system is intuitive (a damped simple harmonic motion) and can be solved analytically, so this system is an ideal test system for our program.



The tetrahedron of state organizes the physical laws that define the behavior of most systems. It plots four variables. The effort (e) and flow (f) are called power variables, because they multiply to give power. Effort corresponds to voltage, force, or torque, and flow corresponds to current, velocity, or angular velocity. The momentum (p) and charge (q) are called state variables, because they indicate the current state of energy storage elements (such as inductors or capacitors).

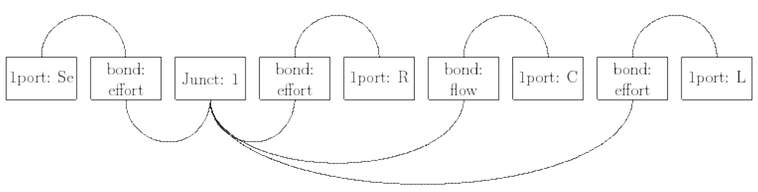


Each branch of the tetrahedron represents a different type of physical component. The left branch represents inductive elements (i.e. masses, electrical inductors, etc.). The center branch represents resistive elements. The right branch represents capacitive elements (i.e. springs, electrical capacitors, etc.). In the left and right branch, a state variable represents the current state and is directly proportional to one power variable. The other power variable is the rate of change of the state variable. In the center branch, the two power variables are directly proportional.



In order to create the visual bond graph representation the program must store locations of mouse-clicks, the movement of the mouse, and the values of key-strokes. These keystrokes and clicks create graphical-node or bond structures that contain a position and a type.
An interpreter analyzes the list of graphical-nodes and bonds to give each a unique index and determine connections between them. The user then changes default values initialized by the program. This list is passed to another interpreter that creates a linear bond graph structure.

```
(list (make-1port    0 'Se 1 9)
      (make-bond     1 0 2 'none)
      (make-junction 2 1  (list 1 3 5 7))
      (make-bond     3 2 4 'none)
      (make-1port    4 'R 3 2)
      (make-bond     5 2 6 'none)
      (make-1port    6 'C 5 5)
      (make-bond     7 2 8 'none)
      (make-1port    8 'L 7 121))
```

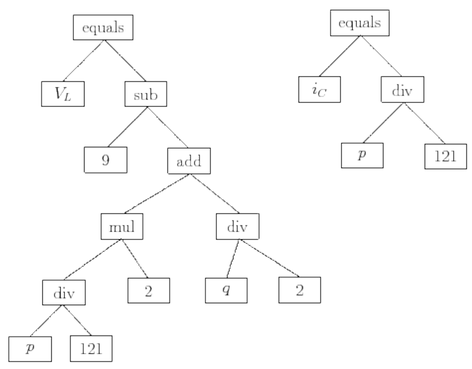


The data structure the GUI sends to the modeling interpreters is a linear representation of a bond graph. Every element in the graph is given an index and refers to the indexes of the connecting elements.
There are four types of element in a graph:

- 1-port elements include sources, resistors, capacitors, and inductors
- 2-port elements include transformers and gyrators
- Junctions connect to an arbitrary number of bonds, and can be 0 or 1 junctions (parallel or series)
- bonds connect two other elements to each other

The initial interpreter proofreads for user error, assignes causality (directions for the equation generator), and then passes the linear bond graph to the equation interpreters.

After a linear bond graph structure is created, a binary equation structure for the controlling equations can be determined.
The behavior of a system is determined by its state variables (the momentum of inductors and charge or position of capacitors). A specific set of values determines the exact behavior of the system at a given point in time.
The mathematical equation defining a variable's change over time is called a differential equation. To model a system over time, a differential equation for each state variable is necessary.
These equations come from the tetrahedron of state. The rate of change of a state variable is a power variable (ex. the rate of change of a spring's position is the velocity).



The equation interpreter works by creating a differential equation for each state variable. It initializes each differential equation to equal a power variable. Using the tetrahedron of state, the interpreter follows the power bonds, substituting new equations for each power variable in the equation. This process continues until the differential equation includes only state variables, constants, and numbers. If necessary, auxiliary equations are generated.
This image shows a visual representation for the equation structure of a RLC circuit. The interpreter starts at the top of the binary tree, and gradually steps through the graph to completion. Each step substitutes a new equation for a power variable, making the binary tree one level deeper.

$$\begin{cases} \dot{p} = V - \left(\frac{R}{L}p + \frac{1}{C}q\right) \\ \dot{q} = \frac{p}{L} \end{cases}$$

$$\Downarrow$$

$$\begin{cases} \dot{p} = 9 - \left(\frac{2}{121}p + \frac{1}{5}q\right) \\ \dot{q} = \frac{p}{121} \end{cases}$$

After all necessary equations are derived as a binary tree structure, they can be rearranged algebraically to a format more useful for an equation solver.
The image to the left shows the simplified differential equations. The dot over a variable indicates its time rate of change or its derivative. This is the ideal format for finding a closed form solution of the system's behavior.
The image to the right shows the differential equations rearranged into matrix form. In a strictly linear system, as many simple systems are, the matrix form conveniently organizes all the variables and coefficients. This format is more useful for a numerical solver because it generalizes calculations.
The interpreter generates the matrix form of binary equation structures by flattening the tree using the distributive property. This creates a list of linear terms. A linear term is composed of a variable expression or a constant. The terms are sorted by variable, and the coefficients and variables are stored in matrix form.
The matrices $\mathbf{X}$ and $\mathbf{U}$ are the two variable matrices. Matrix $\mathbf{X}$ holds the derivative terms and any necessary auxiliary power variables. Matrix $\mathbf{U}$ holds the state variables and has a 1 at the bottom to permit constant terms in the equations.

$$\mathbf{AX} = \mathbf{BU}$$

$$\mathbf{A}\ \mathbf{X} = \mathbf{B}\ \mathbf{U}$$

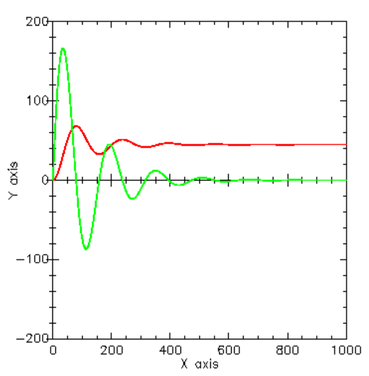
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -\frac{2}{121} & -\frac{1}{5} & 9 \\ \frac{1}{121} & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ q \\ 1 \end{bmatrix}$$

$\mathbf{U}_t$=$\mathbf{U}$ [CurrentStateVariables]
$\mathbf{X}_t$=$\mathbf{C} * \mathbf{U}_t$ [CurrentRatesofChange]
$\Delta\mathbf{U}$=$\Delta t * \mathbf{X}_t$ [ChangeinStateVariables]
$\mathbf{U}_{t+\Delta t}$=$\mathbf{U}_t + \Delta\mathbf{U}$ [NewStateVariables]

When the set of differential equations have been converted to matrix form, they are passed to a numerical solver. This solver uses Gaussian elimination to rearrange the matrix equation to give $\mathbf{X} = \mathbf{CU}$, and then applies the Euler method to generate a list of future values.

The Euler method is an approximation to the future value of a differential equations. The current state variables are substituted into matrix $\mathbf{U}$, and the matrix equation is solved to give immediate values of the rates of change. These rates of change are multiplied by the time-step (a small, discrete increment of time) to give an approximation of how much the state variables change during the time-step. These changes are then added to the state variables to give the next value. In mathematical form, this process can be represented by the loop to the left.

The list of intermediate values of $\mathbf{U}$ is returned to the GUI, which graphs the value of each state variable against time. The Euler method described above is the simplest numeric solver, but is relatively inaccurate. In order to improve accuracy, the Runge-Kutta method was implemented as an alternative solver.



The Runge-Kutta method proved to return a more accurate than the Euler method could. For small time step sizes (ex. $\Delta t = 0.1s$), this difference is negligible, so the Euler method is more useful due to its simpler implementation. With larger time step sizes (ex. $\Delta t = 10s$), the Euler method becomes more inaccurate while the Runge-Kutta method still returns fairly accurate results. Therefore, in uses of our system requiring larger time steps (for example, if the system must be modeled over a much greater period of time), the Runge-Kutta method's accuracy makes it more useful.
Despite its strengths, our system has a few limitations. The first one is graphical, since only numbers can be input as values for circuit elements. Thus, nothing in a system can be time-varying (e.g. only DC circuits can be modeled). The other limitation is that only linear elements can be modeled. Nonlinear elements such as a PN junction (diode) or an alternating current source cannot be modeled. This assumption was made to simplify equation processing and integration. A sufficiently powerful symbolic algebra system can be substituted into our system to resolve this limitation. Finally, due to similarities between various physical domains, our system is capable of modeling complex linear and rotational systems. Other systems, such as hydraulic, can be modeled once our system handles nonlinear components. All that is necessary to model any system is a bond graph representation of it.

λ Viewpoint School 23620 Mulholland Hwy Calabasas, CA 91302 USA