MemSim:
A Membrane Dynamics Simulator

Introduction:

MemSim uses electrical circuit models to emulate the behavior of an elastic membrane subjected to any combination of velocity and force conditions. The user defines the properties of the membrane and the material it is made of; he also specifies source layouts and parameters. Finally, he has control over simulation parameters such as time step and granularity. MemSim requires an FPU or emulator (like the shareware Software FPU), a 68020 or better processor, and at least a portable-sized screen. Color output can be generated on machines with color Quickdraw and system 6.04 or newer. The appearance of such output will be best in 256 color mode.

Source code for the program and/or a version that does not require an FPU (but is significantly slower) will be mailed to any who send me an SASE along with a disk and the $10 shareware fee.

The Model:

The first step in the process of creating a simulator is developing a model for the system of interest. In order to use electrical circuit analysis techniques and intuition, it may also be necessary to translate between engineering disciplines, as was here the case. Since I envisioned most of the velocity constraints on the membrane being essentially boundary conditions and most of the force constraints being imposed by electromechanical transducers, I chose to represent velocity as analogous to voltage and force as analogous to current. Using this mapping, the following equations can be derived:

$$f = m\,a \;\; \dashrightarrow \;\; f = m\,({}^{du}/_{dt}) \;\; \longleftrightarrow \;\; i = C\,({}^{dv}/_{dt})$$

$$f = kx \;\; \dashrightarrow \;\; u = (1/k)\,({}^{df}/_{dt}) \;\; \longleftrightarrow \;\; v = L\,({}^{di}/_{dt})$$

$$u = fR \;\; \longleftrightarrow \;\; v = iR$$

Thus mechanical masses of value m in Kg can be replaced in a circuit model by capacitors with value m in F, mechanical springs with value k in N/m can be replaced by inductors with value (1/k) in H, and mechanical resistances with value R in m/Ns can be replaced by electrical resistances of value R in Ω.

Having determined the appropriate element conversions, it remains to model the mechanical construct itself. A membrane is by nature a distributed system, and as such, two-dimensional versions of any of the analytical techniques discussed in chapter eight of the notes could be used to approximate its transmission . However, it is likely that the response of interest in the case of a membrane is the shape of its surface rather than its terminal characteristics, so a multiple-element uniform lumped model is the most suitable. Given that, it is necessary to determine the connection pattern of the lumped elements.

If the membrane is split up into n equal lumps, each lump will have a mass equal to the total mass of the membrane divided by the number of lumps being created:

$$\frac{\text{(membrane width * membrane length * membrane thickness)}}{\text{number of lumps}}\;\rho = m$$

One terminal of each of these masses is connected to all neighboring lumps by a spring constant, representing the elasticity of the membrane, and a damper, representing the tendency of the membrane to stop vibrating once it has begun. These elements must be in parallel, since a DC force applied to the membrane will not necessarily result in a velocity (i.e. it is possible for all of the force to "flow" through the springs). The second terminal of a mechanical mass is always connected to the inertial frame of reference.

Given the assumption that the spring and damper are both operating in a linear range (probably about some operating point on a nonlinear transfer curve), their values can be easily determined. The spring constant is equal to the area of the spring perpendicular to the axis of deflection, times the modulus of elasticity of the spring material , divided by the length of the spring along the axis of deflection. In terms of the size of the entire membrane, this is:

$$\frac{(\text{membrane thickness} * \text{ membrane width} \perp) / \text{branches along width}}{\text{membrane length} / \text{branches along length}} \; E = k$$

Where E has units of N / m^2. The equation for the resistance of the damper is of the same form, with Ρ, the mechanical resistivity of the material in 1/Ns, replacing E and R, the resistance of the damper, replacing k. Obviously, if the membrane is not square or if the lumps are not square, the spring constant along one axis will be different from that along the other. The unit lump described by this model is shown in Figure 1.

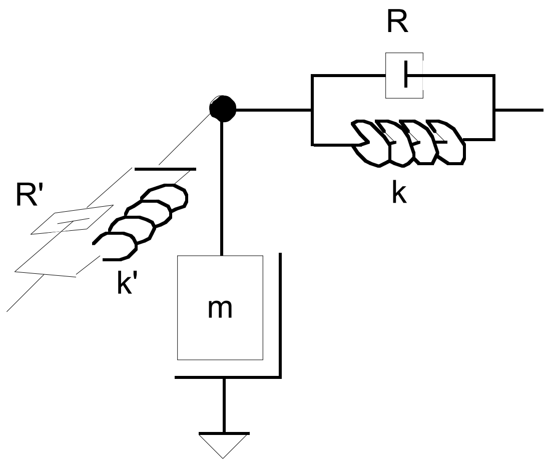


Figure 1: The Unit Element of the Lumped Model

Elements along the left and bottom edges of the membrane have only those branches that connect to neighbors; therefore , the number of branches along an axis is one less than the number of nodes. Force and velocity constraints simply appear as sources between the frame of reference and the appropriate node(s). Using the transformations described above, this mechanical model translates to the electrical model shown in Figure 2.

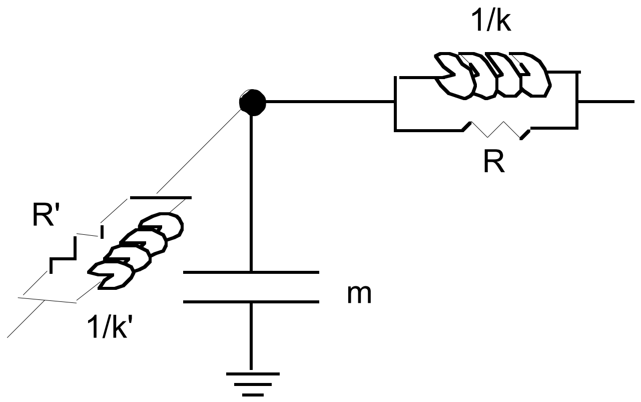


Figure 2: The Electrical Model of a Node

Simulation Plan:

Given these models of the lumped element array, it is necessary to decide on a means of computing all relevant circuit parameters (i.e. state variables V and I or F and U, and from them the position x). There are two steps that must be taken in such a time-varying simulation. At each time step, the correct state variable values must be calculated. Then, between time steps, some type of integrator must perform the integration embodied in the capacitors and inductors. Any time-varying sources must also be updated. My program uses Gauss-Seidel iteration with over-relaxation to compute the state variables at each time step, and uses trapezoidal rule companion circuits to perform the integration between time steps. A procedure also updates all sources in between time steps.

The Gauss-Seidel iteration step is derived from KCL at each node as shown in Figure 3, where the inductors and capacitors have been replaced by their companion circuits. With the possible addition of a force source, I_s, or a velocity source, V_s:

If V_s is defined: $V = V_s$. Otherwise, do KCL:

$$0 = (V_u - V) / R_u + I_u + (V_r - V) / R_r - I_r + (V_b - V) / R_b - I_b + (V_l - V) / R_l + I_l + (I_s) + (V_c - V) / Rc$$

$$V = \frac{(V_u / R_u) + (V_r / R_r) + (V_d / R_d) + (V_l / R_l) + (V_c / R_c) + I_u - I_r - I_d + I_l + (I_s)}{(1 / R_u) + (1 / R_r) + (1 / R_d) + (1 / R_l) + (1 / R_c)}$$

Using over-relaxation makes this:

$V = V_{old} + \text{weight}\ (V_{\text{from above}} - V_{old})$.

The program iterates at each time step using an auto-adjusting weight factor. It keeps a moving average of the number of iterations the last three time steps took to converge, and increases the weight if the current iteration did better than the average. If not, it lowers the weight factor. The average is initialized to a large number to start things in motion, and a protection threshold is set to keep fluctuations in convergence speed from causing an unstable weight to be used (see the functions "converge" and "step"). In between time steps a function updates the companion circuit models for the capacitors and inductors.

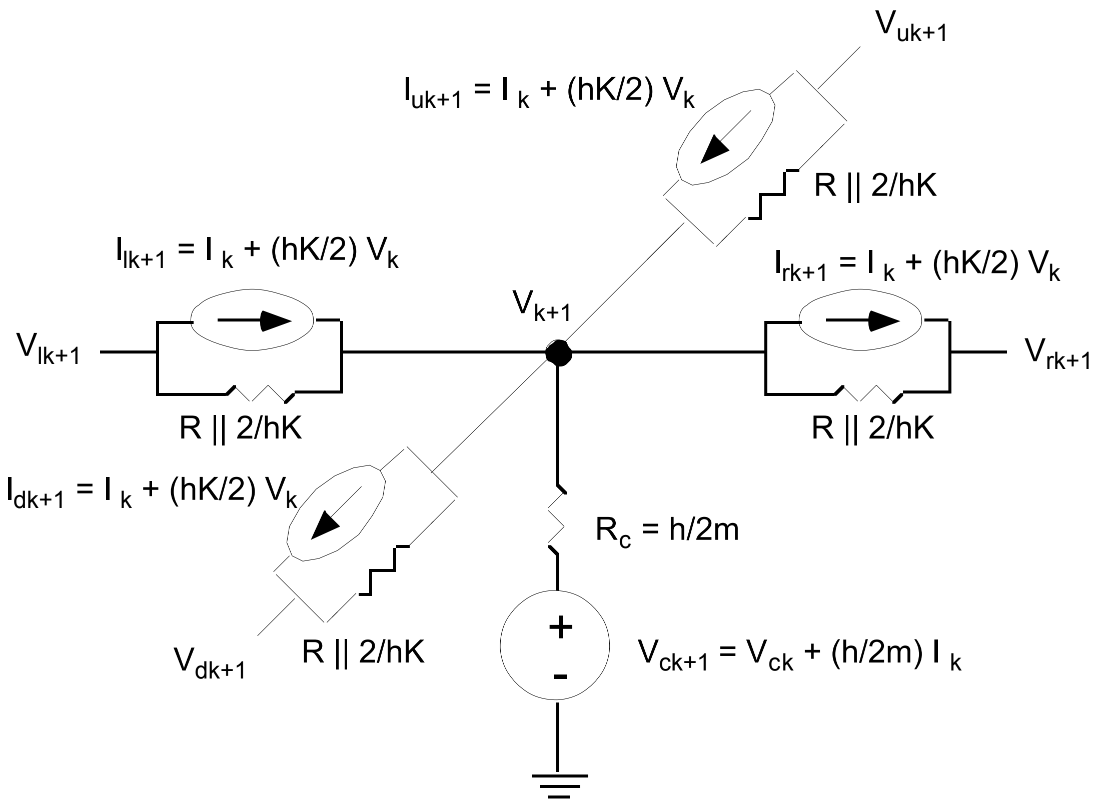

V_{uk+1}
$I_{uk+1} = I_k + (hK/2) V_k$
R || 2/hK
$I_{lk+1} = I_k + (hK/2) V_k$
$I_{rk+1} = I_k + (hK/2) V_k$
V_{k+1}
V_{lk+1}
V_{rk+1}
R || 2/hK
R || 2/hK
$I_{dk+1} = I_k + (hK/2) V_k$
$R_c = h/2m$
+
-
V_{dk+1}
R || 2/hK
$V_{ck+1} = V_{ck} + (h/2m) I_k$

Figure 3: A Node For Iteration / Integration

Sources:

The program provides for graphical entry of source locations and supports the source functions step, impulse, sin, and cos for both force and velocity sources. Sources can also have their own material properties, as would be the case if, for example, a thick wire attached to the back of the membrane were the source of a force excitation. Such a source would cause local variations in spring constant, mass, and resistance. Force source magnitudes are automatically scaled by 1 / (area of source), velocity sources are not.

The graphical source window displays which nodes will be touched by a source - it automatically breaks itself up into a grid the same size as the node space. You may either draw sources in the window or enter them in the text boxes below it; in either case, the set of points that show up in the source box is the set of points that will act as sources on the membrane. Mapping a continuous membrane to a small node space and then to a larger pixel space inevitably causes some inaccuracy. If a source that should fit on the membrane does not show up in the graphical source entry window, it has been lost in the mapping; use a different one.

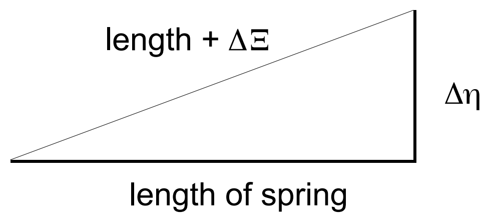
Output:

Once the program has finished its iteration for a given time step, it sends its data to plot routines that create a projected 3-D image of the membrane in a window on the screen. The user can choose not to plot every set of data if he so chooses; this speeds up program execution but makes it harder to follow the pattern of the membrane's motion. Time step and total simulation length are also selectable. The program makes a rough estimate of the lowest natural frequencies of the membrane (simply 1/sqrt(LC)) in order to give the user an idea of what time step is appropriate.

As the simulation proceeds, the user can choose whether to plot the velocities of the masses or the position of the surface. The velocities are simply the voltages at each node; the position is calculated by finding the displacement of each spring relative to its neighbors. The current through the model inductor is the equivalent of the force acting on the mechanical spring, so dividing by the spring constant yields a displacement. The program makes the approximation that the change in length of the spring is all in the up-down direction. Although

one could calculate the up-down displacement by finding the unknown leg of a right triangle (Figure 4), that method is relatively computation intensive;

furthermore, it is unclear what to do if ΔX < 0, that is, if the spring is in compression.



Δη = σθρτ((λεν+ΔΞ)^2 - λεν^2)

Φιγυρε 4: ςερτιχαλ Δισπλαχεμεντ

A plot of ΔH vs. ΔX and length (Figure 5) shows that the approximation is relatively accurate - ΔH ≈ ΔX over a wide range of values of length and ΔX. Outside of the region where the approximation holds, the other assumptions made by the program, such as linear spring action and liner resistance probably fail as well, so it really is not a crucial issue.

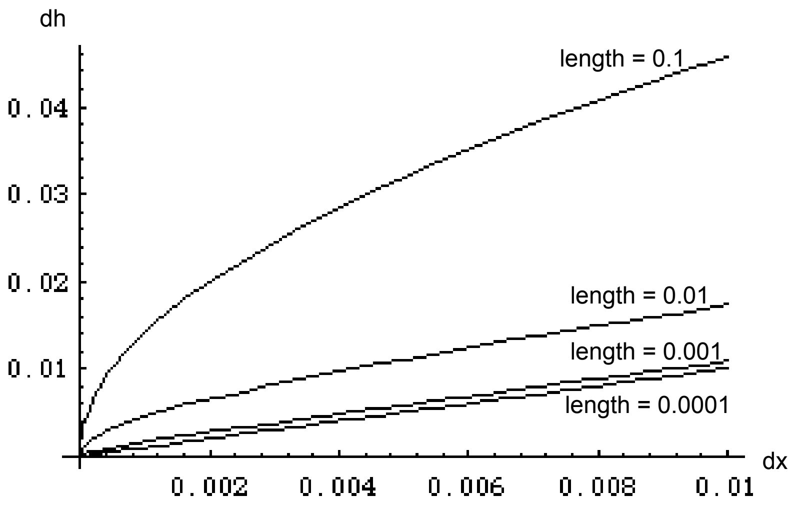


Figure 5: dh vs dx for several lengths

In any case, once the x,y,z space corresponding to the membrane and its up and down position/velocity has been constructed, it is scaled, shifted, rotated, and projected so as to map onto the screen. The values on the z axis will scale automatically if the membrane displacement or velocity exceeds half of the membrane's width, so it the values there seem not to be changing, look at the axis label. If the computer in use supports color Quickdraw, the membrane surface can also be colored to indicate elevation/speed using one of four palettes.

Additional Notes:

This document provides an explanation of the simulation mechanics of the program. Other aspects, such as menu selections, file options, etc. follow standard Macintosh style; it should be easy to find one's way around the program.   MemSim should run on any Macintosh with a 68020 and FPU (or emulator, like the shareware Software FPU) or better. Dialog and window appearances and placement will be best under System 7 in 256 color mode. If the program cannot perform a system check (using Gestalt, System 6.04 or newer), it will display a warning that a 68020 or better and an FPU are required; color will be disabled. Have a look at the example settings - some of them are pretty spiffy (though not very realistic - note the huge deflections in "2D trans").

Defaults:

The default settings for material characteristics were chosen so as to provide behavior that seemed consistent with what one would consider an "elastic" membrane. The modulus of elasticity is about that of rubber, the density should be close as well (after all, some rubber float and some sinks). The resistivity was selected only by the amount of ringing it allowed - it is not based on any values from the references. Raising this number increases the tendency of the membrane to ring, lowering it makes the membrane less mobile. Below a certain point the membrane will not move at all.